Installation guide for Autotrace
-*- text -*-

Index of this file:

1. Basic Installation

These are generic installation instructions.

The `configure' shell script attempts to guess correct values for various system-dependent variables used during compilation. It uses those values to create a `Makefile' in each directory of the package. It may also create one or more `.h' files containing system-dependent definitions. Finally, it creates a shell script `config.status' that you can run in the future to recreate the current configuration, a file `config.cache' that saves the results of its tests to speed up reconfiguring, and a file `config.log' containing compiler output (useful mainly for debugging `configure').

If you need to do unusual things to compile the package, please try to figure out how `configure' could check whether to do them, and mail diffs or instructions to the address given in the `README' so they can be considered for the next release. If at some point `config.cache' contains results you don't want to keep, you may remove or edit it.

The file `configure.in' is used to create `configure' by a program called `autoconf'. You only need `configure.in' if you want to change it or regenerate `configure' using a newer version of `autoconf'.

The simplest way to compile this package is:

1. `cd' to the directory containing the package's source code and type `./configure' to configure the package for your system. If you're using `csh' on an old version of System V, you might need to type `sh ./configure' instead to prevent `csh' from trying to execute `configure' itself.

   Running `configure' takes awhile. While running, it prints some messages telling which features it is checking for.

2. Type `make' to compile the package.

3. Optionally, type `make check' to run any self-tests that come with the package.

4. Type `make install' to install the programs and any data files and documentation.

5. You can remove the program binaries and object files from the source code directory by typing `make clean'. To also remove the

files that `configure' created (so you can compile the package for a different kind of computer), type `make distclean'. There is also a `make maintainer-clean' target, but that is intended mainly for the package's developers. If you use it, you may have to get all sorts of other programs in order to regenerate files that came with the distribution.

Compilers and Options

Some systems require unusual options for compilation or linking that the `configure' script does not know about. You can give `configure' initial values for variables by setting them in the environment. Using a Bourne-compatible shell, you can do that on the command line like this:

```
CC=c89 CFLAGS=-O2 LIBS=-lposix ./configure
```

Or on systems that have the `env' program, you can do it like this:

```
env CPPFLAGS=-I/usr/local/include LDFLAGS=-s ./configure
```

Compiling For Multiple Architectures

You can compile the package for more than one kind of computer at the same time, by placing the object files for each architecture in their own directory. To do this, you must use a version of `make' that supports the `VPATH' variable, such as GNU `make'. `cd' to the directory where you want the object files and executables to go and run the `configure' script. `configure' automatically checks for the source code in the directory that `configure' is in and in `..'.

If you have to use a `make' that does not supports the `VPATH' variable, you have to compile the package for one architecture at a time in the source code directory. After you have installed the package for one architecture, use `make distclean' before reconfiguring for another architecture.

Installation Names

By default, `make install' will install the package's files in `/usr/local/bin', `/usr/local/man', etc. You can specify an installation prefix other than `/usr/local' by giving `configure' the option `--prefix=PATH'.

You can specify separate installation prefixes for architecture-specific files and architecture-independent files. If you give `configure' the option `--exec-prefix=PATH', the package will use PATH as the prefix for installing programs and libraries. Documentation and other data files will still use the regular prefix.

In addition, if you use an unusual directory layout you can give options like `--bindir=PATH' to specify different values for particular kinds of files. Run `configure --help' for a list of the directories

you can set and what kinds of files go in them.

If the package supports it, you can cause programs to be installed with an extra prefix or suffix on their names by giving `configure' the option `--program-prefix=PREFIX' or `--program-suffix=SUFFIX'.

Optional Features

Some packages pay attention to `--enable-FEATURE' options to `configure', where FEATURE indicates an optional part of the package. They may also pay attention to `--with-PACKAGE' options, where PACKAGE is something like `gnu-as' or `x' (for the X Window System). The `README' should mention any `--enable-' and `--with-' options that the package recognizes.

For packages that use the X Window System, `configure' can usually find the X include and library files automatically, but if it doesn't, you can use the `configure' options `--x-includes=DIR' and `--x-libraries=DIR' to specify their locations.

Specifying the System Type

There may be some features `configure' can not figure out automatically, but needs to determine by the type of host the package will run on. Usually `configure' can figure that out, but if it prints a message saying it can not guess the host type, give it the `--host=TYPE' option. TYPE can either be a short name for the system type, such as `sun4', or a canonical name with three fields:
     CPU-COMPANY-SYSTEM

See the file `config.sub' for the possible values of each field. If `config.sub' isn't included in this package, then this package doesn't need to know the host type.

If you are building compiler tools for cross-compiling, you can also use the `--target=TYPE' option to select the type of system they will produce code for and the `--build=TYPE' option to select the type of system on which you are compiling the package.

Sharing Defaults

If you want to set default values for `configure' scripts to share, you can create a site shell script called `config.site' that gives default values for variables like `CC', `cache_file', and `prefix'. `configure' looks for `PREFIX/share/config.site' if it exists, then `PREFIX/etc/config.site' if it exists. Or, you can set the `CONFIG_SITE' environment variable to the location of the site script. A warning: not all `configure' scripts look for a site script.

Operation Controls

`configure' recognizes the following options to control how it operates.

`--cache-file=FILE'
Use and save the results of the tests in FILE instead of `./config.cache'. Set FILE to `/dev/null' to disable caching, for debugging `configure'.

`--help'
Print a summary of the options to `configure', and exit.

`--quiet'
`--silent'
`-q'
Do not print messages saying which checks are being made. To suppress all normal output, redirect it to `/dev/null' (any error messages will still be shown).

`--srcdir=DIR'
Look for the package's source code in directory DIR. Usually `configure' can determine that directory automatically.

`--version'
Print the version of Autoconf used to generate the `configure' script, and exit.

`configure' also accepts some other, not widely useful, options.

2. ImageMagick related issues

If you use Red Hat Linux 7.2 to build autotrace with ImageMagick rpm, you will get a trouble something like:

```
/bin/sh ./libtool --mode=link gcc  -g -O2  -o autotrace  atou.o main.o ...
libtool: link: cannot find the library `/usr/lib/libxml2.la'
gmake: *** [autotrace] Error 1
```

The reasons of thie trouble are 1. libMagick.la is broken; and 2. libxml2.so does not exist. There are two ways to avoid this trouble.

1. Build autotrace without ImageMagick.
   Run configure with "--withoud-magick" option(then run make clean; make).
   However, you lost input functions that use ImageMagick.

2. Hack the broken files.
   Replace "/usr/lib/libxml2.la" with -lxml2 in ImageMagick.la then
   Make a symbolic link, /usr/lib/libxml2.so, that referees /usr/lib/libxml2.so.2.
   If you don't understand what I write, you should not do.

If you use Red Hat Linux 8.0 to build autotrace with ImageMagick rpm, you will get a trouble something like in configure time:

```
    checking for Magick-config... Magick-config
    checking magick/api.h usability... no
    checking magick/api.h presence... no
    checking for magick/api.h... no
    configure: WARNING: *** Magick-config is found but magick/api.h is
not found in -I/usr/X11R6/include/X11/magick -D_REENTRANT -
D_FILE_OFFSET_BITS=64 -I/usr/X11R6/include -I/usr/X11R6/include/X11 -
I/usr/include/freetype2 -I/usr/include/libxml2 ***
    configure: WARNING: *** Check Magick-config.
***
    configure: WARNING: *** ImageMagick input handler is disabled.
***
```

in spite of following rpm -q results:

```
    [jet@chuf autotrace]$ rpm -q ImageMagick
    ImageMagick-5.4.7-5
    [jet@chuf autotrace]$ rpm -q ImageMagick-devel
    ImageMagick-devel-5.4.7-5
```

I guess some of header files are missed in ImageMagick-devel.
I recommend you to install ImageMagick from tar.gz. file.

Pstoedit related issues

If you are using pstoedit-3.32, pstoedit checking in configure of autotrace runs TWICE. Ignore the warning message of first checking.
If you are using pstoedit-3.33 or higher, pstoedit checking runs only once. If you got a trouble to build autotrace with pstoedit, you can disable to use pstoedit with giving --without-pstoedit to configure of autotrace. Is you want to use pstoedit anyway linked with autotrace, let me(Masatake YAMATO<yamato@redhat.com> or autotrace mailing list) know following informations:

- your operating system name and version
- autotrace version
- pstoedit version
- value of $LD_LIBRARY_PATH
- value of $PATH
- options given to configure of autotrace
- options given to configure of pstoedit
- output of configure of autotrace
- output of configure of pstoedit
- config.log of configure of autotrace
- config.log of configure of pstoedit
- Makefile of autotrace if generated
- src/Makefile of pstoedit
- output of pstoedit-config --libs

output of pstoedit-config --cflags
output of autotrace-config --libs
output of autotrace-config --cflags
path for pstoedit-config
path for autotrace-config
path for libautotrace.so
path for libpstoedit.so
if there is a compile farm on that the same OS as you use

(Of couse, I(Masatake YAMATO) cannot promise anything even
if you sent me above informations.)