\* The most simple case

-----------------------

If you compile wxWidgets on Linux for the first time and don't like to read

install instructions just do (in the base dir):

> ./configure --with-wine

> make

> su <type root password>

> make install

> ldconfig

> exit

On all variants of Unix except Linux (and maybe except \*BSD), shared libraries

are not supported out of the box due to the utter stupidity of libtool, so you'll

have to do this to get shared library support:

> ./configure --with-wine --disable-unicode --disable-static --enable-shared

Then you'll have to edit the wrongly created libtool script. There are two

important entries with respect to shared library creation, which are

archive\_cmds="\$LD -shared ....

archive\_expsym\_cmds="\$LD -shared ....

which should be something like

archive\_cmds="\$CC -shared ....

archive\_expsym\_cmds="\$CC -shared ....

Afterwards you can continue with

> make

> su <type root password>

> make install

> ldconfig

> exit

If you want to remove wxWidgets on Unix you can do this:

> su <type root password>

> make uninstall

> ldconfig

> exit

\* The expert case

-----------------

If you want to do some more serious cross-platform programming with wxWidgets,

such as for GTK and Motif, you can now build two complete libraries and use

them concurrently. For this end, you have to create a directory for each build

of wxWidgets - you may also want to create different versions of wxWidgets

and test them concurrently. Most typically, this would be a version configured

with --enable-debug\_flag and one without. Note, that only one build can currently

be installed, so you'd have to use local version of the library for that purpose.

For building three versions (one GTK, one WINE and a debug version of the WINE

source) you'd do this:

md buildmotif

cd buildmotif

../configure --with-motif

make

cd ..

md buildwine

cd buildwine

../configure --with-wine

make

cd ..

md buildwined

cd buildwined

../configure --with-wine --enable-debug\_flag

make

cd ..

\* The most simple errors

------------------------

wxWINE doesn't work yet as WINE isn't really up to the task yet.

You get errors during compilation: The reason is that you probably have a broken

compiler, which includes almost everything that is called gcc. If you use gcc 2.8

you have to disable optimisation as the compiler will give up with an internal

compiler error.

If there is just any way for you to use egcs, use egcs. We cannot fix gcc.

You get immediate segfault when starting any sample or application: This is either

due to having compiled the library with different flags or options than your program -

typically you might have the \_\_WXDEBUG\_\_ option set for the library but not for your

program - or due to using a broken compiler (and its optimisation) such as GCC 2.8.

\* The most simple program

-------------------------

Now create your super-application myfoo.app and compile anywhere with

g++ myfoo.cpp `wx-config --libs --cflags` -o myfoo

\* General

-----------------------

The Unix variants of wxWidgets use GNU configure. If you have problems with your

make use GNU make instead.

If you have general problems with installation, read my homepage at

http://wesley.informatik.uni-freiburg.de/~wxxt

for newest information. If you still don't have any success, please send a bug

report to one of our mailing lists (see my homepage) INCLUDING A DESCRIPTION OF

YOUR SYSTEM AND YOUR PROBLEM, SUCH AS YOUR VERSION OF WINE, WXWINE, WHAT DISTRIBUTION

YOU USE AND WHAT ERROR WAS REPORTED. I know this has no effect, but I tried...

\* GUI libraries

-----------------------

wxWidgets/WINE requires the WINE library to be installed on your system.

You can get the newest version of the WINE from the WINE homepage at:

http://www.winehq.com

\* Create your configuration

-----------------------------

Usage:

./configure options

If you want to use system's C and C++ compiler,

set environment variables CC and CCC as

% setenv CC cc

% setenv CCC CC

% ./configure options

to see all the options please use:

./configure --help

The basic philosophy is that if you want to use different

configurations, like a debug and a release version,

or use the same source tree on different systems,

you have only to change the environment variable OSTYPE.

(Sadly this variable is not set by default on some systems

in some shells - on SGI's for example). So you will have to

set it there. This variable HAS to be set before starting

configure, so that it knows which system it tries to

configure for.

Configure will complain if the system variable OSTYPE has

not been defined. And Make in some circumstances as well...

\* General options

-------------------

Given below are the commands to change the default behaviour,

i.e. if it says "--disable-threads" it means that threads

are enabled by default.

Many of the configure options have been thoroughly tested

in wxWidgets snapshot 6, but not yet all (ODBC not).

You must do this by running configure with either of:

--with-wine Use the WINE library

The following options handle the kind of library you want to build.

--enable-threads Compile with thread support. Threads

support is also required for the

socket code to work.

--disable-shared Do not create shared libraries.

--disable-optimise Do not optimise the code. Can

sometimes be useful for debugging

and is required on some architectures

such as Sun with gcc 2.8.X which

would otherwise produce segvs.

--enable-profile Add profiling info to the object

files. Currently broken, I think.

--enable-no\_rtti Enable compilation without creation of

C++ RTTI information in object files.

This will speed-up compilation and reduce

binary size.

--enable-no\_exceptions Enable compilation without creation of

C++ exception information in object files.

This will speed-up compilation and reduce

binary size. Also fewer crashes during the

actual compilation...

--enable-mem\_tracing Add built-in memory tracing.

--enable-dmalloc Use the dmalloc memory debugger.

Read more at www.letters.com/dmalloc/

--enable-debug\_info Add debug info to object files and

executables for use with debuggers

such as gdb (or its many frontends).

--enable-debug\_flag Define \_\_DEBUG\_\_ and \_\_WXDEBUG\_\_ when

compiling. This enable wxWidgets' very

useful internal debugging tricks (such

as automatically reporting illegal calls)

to work. Note that program and library

must be compiled with the same debug

options.

\* Feature Options

-------------------

Many of the configure options have been thoroughly tested

in wxWidgets snapshot 6, but not yet all (ODBC not).

When producing an executable that is linked statically with wxGTK

you'll be surprised at its immense size. This can sometimes be

drastically reduced by removing features from wxWidgets that

are not used in your program. The most relevant such features

are

--without-libpng Disables PNG image format code.

--without-libjpeg Disables JPEG image format code.

{ --without-odbc Disables ODBC code. Not yet. }

--disable-resources Disables the use of \*.wxr type

resources.

--disable-threads Disables threads. Will also

disable sockets.

--disable-sockets Disables sockets.

--disable-dnd Disables Drag'n'Drop.

--disable-clipboard Disables Clipboard.

--disable-serial Disables object instance serialisation.

--disable-streams Disables the wxStream classes.

--disable-file Disables the wxFile class.

--disable-textfile Disables the wxTextFile class.

--disable-intl Disables the internationalisation.

--disable-validators Disables validators.

--disable-accel Disables accel.

Apart from disabling certain features you can very often "strip"

the program of its debugging information resulting in a significant

reduction in size.

\* Compiling

-------------

The following must be done in the base directory (e.g. ~/wxGTK

or ~/wxWin or whatever)

Now the makefiles are created (by configure) and you can compile

the library by typing:

make

make yourself some coffee, as it will take some time. On an old

386SX possibly two weeks. During compilation, you'll get a few

warning messages depending in your compiler.

If you want to be more selective, you can change into a specific

directory and type "make" there.

Then you may install the library and its header files under

/usr/local/include/wx and /usr/local/lib respectively. You

have to log in as root (i.e. run "su" and enter the root

password) and type

make install

You can remove any traces of wxWidgets by typing

make uninstall

If you want to save disk space by removing unnecessary

object-files:

make clean

in the various directories will do the work for you.

\* Creating a new Project

--------------------------

1) The first way uses the installed libraries and header files

automatically using wx-config

g++ myfoo.cpp `wx-config --libs` `wx-config --cflags` -o myfoo

Using this way, a make file for the minimal sample would look

like this

CXX = g++

minimal: minimal.o

$(CXX) -o minimal minimal.o `wx-config --libs`

minimal.o: minimal.cpp

$(CXX) `wx-config --cflags` -c minimal.cpp -o minimal.o

clean:

rm -f \*.o minimal

This is certain to become the standard way unless we decide

to stick to tmake.

2) The other way creates a project within the source code

directories of wxWidgets. For this endeavour, you'll need

the usual number of GNU tools, at least

GNU automake version 1.4

GNU autoheader version 2.14

GNU autoconf version 2.14

GNU libtool version 1.3

and quite possibly

GNU make

GNU C++

and if you have all this then you probably know enough to

go ahead yourself :-)

----------------------

In the hope that it will be useful,

Robert Roebling <roebling@sun2.ruf.uni-freiburg.de>

Addition notes by Julian Smart, August 2002

===========================================

I've fixed some compile errors, and got as far as

compiling wxWINE, but actually linking a sample will take

further work.

To compile wxWINE, export these variables:

export CPPFLAGS=-I/usr/local/include/wine

export LDFLAGS=-L/usr/local/lib/wine

and configure with:

configure --disable-static --enable-shared --enable-gui \

--with-wine --without-libpng --enable-debug\_flag --enable-log \

--enable-debug\_info --enable-ole --enable-clipboard --enable-dataobj \

--enable-debug --enable-threads --disable-sockets \

--with-libjpeg --enable-debug\_cntxt

Compiling a sample won't work yet because 'winebuild' needs

to be called, and the resuling C file compiled and linked.

Plus, Windows DLLs need to be imported.

Note that the documentation on the WINE web site on using

winebuild is out of date (August 2002) -- the spec file no

longer supports import and type keywords. Instead look at

samples in the WINE 'programs' directory for inspiration

and compile options to use. It's probable that the

wxWINE library will need recompiling with different options.

Any progress on this front will be very welcome.

Note that while wxWINE builds with --enable-unicode, samples

don't run. Some samples will run when built with

--disable-unicode, and others (such as auidemo) fail.