wxWindows Frequently Asked Questions Version 1.7

Julian Smart and others

October 1997

Contents

1. About this document

This is the questions-and-answers document for wxWindows, the free multi-platform GUI C++ library. Please feel free to comment on this FAQ and submit new entries.

2. General questions

2.1. What are the licensing considerations for wxWindows?

None. You may make any use of any part of wxWindows, and no payment is necessary. While it's nice if you acknowledge the author(s) of wxWindows in your own work, it's not necessary.

Correspondingly, there is no warranty with wxWindows, as you have probably guessed by now.

2.2. Why is wxWindows free, and will it ever be commercial or shareware?

It's free because:

- it was never intended to compete as a commercial product, since there are not sufficient resources at AIAI.
- the feedback and bug fixes that AIAI get from the Internet community are extremely valuable.
- having used much free software in the past, it's only right to put something back.
- AIAI gains a small amount of publicity (well, so the story goes).
- I got bored of writing code that never saw the light of day.

Neither I nor AIAI have any plans to produce a commercial or shareware version.

2.3. Why the silly name?

w for MS Windows, **x** for the X windowing system, Windows for those rectangular things you see a lot of. Ok, so it's not exactly inspired.

2.4. What bitmap loading facilities are available for wxWindows?

There is Windows .BMP code that compiles under Windows distributed with wxWindows 1.50k (utils/dib directory). wxWindows 1.60 will allow proper colourmap setting with this code. DIB allows loading and saving BMP files. The contributed Windows wxImage library (utils/wximage/win) enables GIF loading (and other formats such as JPEG if extra graphics libraries are installed).

For X, the utils/image directory contains code to load GIFS, Windows bitmaps and X bitmaps into a canvas (and optionally, into a wxBitmap). The code has been taken from a pre-shareware version of the excellent image viewer XV; it cannot be guaranteed that the code is free from copyright issues. See the file test.cpp for a scanty explanation of how to use it.

XPM (colour X Pixmap) files are supported by the wxXPM package now bundled with wxWindows, from 1.61. The package is in contrib/wxxpm, and includes a utility XPMShow which allows conversion between XPM and BMP files under Windows (only, at present).

For the maximum bitmap facilities, wx_setup.h should be edited and the following settings made:

- Set USE_IMAGE_LOADING_IN_X to 1
- Set USE_IMAGE_LOADING_IN_MSW to 1

- Set USE_XPM_IN_X to 1
- Set USE_XPM_IN_MSW to 1

Then recompile the wxWindows, image (X), DIB (Windows) and XPM (X and Windows) libraries and link in with your application. You can now load and save XPMs (X and Windows), load BMP files (X and Windows), save BMP bitmaps (Windows), and load GIFs (X), all through the wxBitmap interface.

It is recommended that you use XPMs for colour bitmap buttons, at least under X. Note that colour buttons will not display correctly on X terminals whose display depth does not match the bitmap depth, so checking will need to be done in the application.

2.5. How can I convert Windows bitmaps to UNIX formats?

To convert Windows bitmaps to XBM (monochrome X bitmap), you can use the shareware package PaintShop Pro to change coloured areas to black or white, and then save the image as a PPM file. On UNIX, use the image manipulation package XV to convert from PPM to XBM.

You can convert to XPM using XPMShow, in the wxWindos `utils/xpmshow` directory.

2.6. Can I put a canvas (or text subwindow) in a panel?

Before 1.61, this kind of nesting was not allowed, because XView doesn't support it, and wxWindows was heavily influenced by XView.

From 1.61, such restrictions are being relaxed a bit for platforms that support more flexibility. Under Motif and Windows, canvases and text subwindows can be placed in panels as well as in frames.

Also, again from 1.61 on, wxPanel is a subclass of wxCanvas, under Windows (only, at present). So drawing in panels is now possible (or placing panel items in a canvas, whichever way you like to look at it). Hopefully this will soon be extended to Motif; it is unlikely to be implemented for XView since XView doesn't really support this way of working.

2.7. Can I draw in a panel, or place panel items in a canvas?

See *Nesting subwindows* (page 3).

2.8. How were the samples (and other code) created?

The samples and other code were all created with a text editor, with nary an Integrated Development Environment in sight. In future, as wxBuilder and the wxWindows resource system matures, it is to be hoped that some code will have been generated by wxBuilder.

See also *Can I use an IDE?* (page 27).

2.9. How can I use more than the basic Windows colours in a 256-colour mode graphics adapter?

Andrew Davison has the following advice for creating a colourmap. Once you have a valid wxColourMap, you need to set the colourmap for the device context and window.

```
At 12:13 5/06/96 -0400, you wrote:
>This is what I did to set the wxColourMap :
>
>     const int nocol = 256 ;
>
>     unsigned char red[nocol] ;
>     unsigned char blue[nocol] ;
>     unsigned char green[nocol] ;
>
>     int i ;
>     for (i=0 ; i<nocol ; i++){
>           red[i] = i ;
>           green[i] = i ;
>           blue[i] = i ;
>     }
>
>     // wxCM is a wxColourMap
>     wxCM.Create(nocol, red, green, blue) ;
>}
>
>What I do after is wxDC::SetColourMap(&wxCM) with the wxDC's I use.
>
>I'm not to sure with what values to fill the arrays I pass to
::Create.
>With these values (i.. [0..255]) it didn't change anything.
>
>     If anyone has a clue,
>                       Patrick

I would suggest modelling the Netscape colour-cube, which attempts
to evenly divide up the spectrum.

 Netscape uses only 216 colours plus another 4 for it's logo. How you
fill
in the rest is up to you, but under under MSW Windows itself will
always steal
20 colors.

Select colours as follows and load them into your wxColourMap. Using
r, g and b values of 0x00, 0x33, 0x66, 0x99, 0xCC, 0xFF respectively.

i.e

  unsigned char r[256],
  unsigned i = 0;

  for (int r = 0x00; r <= 0xFF; r += 0x33)
      for (int g = 0x00; g <= 0xFF; g += 0x33)
          for (int b = 0x00; b <= 0xFF; b += 0x33)
          {
              red[i] = r;
              green[i] = g;
              blue[i] = b;
              i++;
          }

Given the above scenario it's always likely that your randomly chosen
```

RGB value
will have a close match.

It would be nice of course if WXWINDOWS detected a 256 colour adapter and did
something similar.

Regards.

Andrew Davison

3. Compilation issues

3.1. General

3.1.1. What I can do to reduce executable size?

wxWindows does produce large executables, but there's quite a lot one can do to mitigate the problems.

- Eliminate debug info, either at link time or compile time. The latter usually has a slight size advantage over the former, but you may consider it not worth the hassle of completely recompiling for delivery.
- Optimise for space. Depending on the compiler, this can be quite significant. With some compilers you can't optimise because of compiler bugs. However, when I started using Watcom C++ to produce WIN32 executables, I got a reduction from 3MB for my main tool (Hardy) compiled under VC++ 1.5, to 2MB under Watcom, with optimisation. This is actually not a bad size for a piece of software that has been growing for several years, and does quite a lot.
- Eliminate unnecessary modules in wxWindows through wx_setup.h and makefile settings, e.g. wxXPM, doc/view, wxPostScriptDC. Through doing this, my WIN32 Tex2RTF LaTeX to HTML, WinHelp/RTF and wxHelp converter is 633 KB (compiled with Watcom). Quite a reasonable size these days.
- Compress the executable with a suitable compressor (that can make a self-extracting .exe, transparent to the user), or other method for distribution. I don't do this myself, but it would be a neat way of saving disk space.
- On UNIX, create a shared version of the wxWindows library; see contrib/wxshlib for GNU autoconf files for building shared and static libraries.
- Under Windows, make a DLL out of wxWindows: not yet possible, but will probably be done in the next few months because WIN32 DLLs are apparently easier to make than WIN16 ones. This will spectacularly reduce the size of executables, at the cost of confusion with DLL versions. But for saving disk space during development this would be a good option, and will probably reduce link times too. See *Making a Windows DLL* (page 36) for notes on a not-quite-successful attempt to make a DLL under VC++ 4.0.
- Get a better optimising linker. Part of the size problem is that the compilers don't throw away sufficient bits of unused code. They may get better at doing this as time goes on and everyone's executables get larger.

The Windows hello demo executable is around 700KB for VC++ 1.5, less for Watcom. This demo differs from traditional hello demos in using quite a lot of GUI functionality, and therefore pulling in a lot of code. And most options are enabled in wx_setup.h.

Obviously, the larger your application, the less you notice the overhead of wxWindows.

3.1.2. What are ItsyBits, FAFA etc.? Which libraries do I really need to compile?

From wxWindows 1.61, the makefiles have been altered so that several 'subordinate' libraries are compiled into wx.lib (or wx_motif.a or whatever). This means that configuration of wxWindows is much more centralized, and it's not necessary to fiddle with many makefiles if you decide to compile in a specific wxWindows feature.

These little libraries add optional functionality to wxWindows, supported in the wxWindows class

library but the bulk of the functionality being implemented separately for modularity (and potential copyright) reasons.

Unfortunately, you do have to edit *both* wx_setup.h and the makefile in src/x or src/msw in order to configure wxWindows. So it may be easier to compile all libraries rather than try to configure wxWindows, unless you're really having trouble compiling one of the libraries.

Here's a list of the optional libraries (found in wx/contrib or wx/utils). The relevant wx_setup.h identifier is given in brackets.

CTL3D Windows only: allows use of 3D style controls (CTL3D).

FAFA Windows only: allows use of bitmap buttons, messages and radiobuttons (FAFA_LIB).

ItsyBitsy Windows only: supports tiny titlebars (USE_ITSY_BITSY).

Gauge Windows only: necessary for implementation of wxGauge class (USE_GAUGE).

xmGauge Motif only: necessary for implementation of wxGauge class (USE_GAUGE).

wxXPM All platforms: necessary for implementation of XPM pixmap functionality (USE_XPM_IN_X, USE_XPM_IN_MSW).

DIB Windows only: necessary for implementation of BMP loading/saving functionality (USE_IMAGE_LOADING_IN_MSW).

wxImage X only: necessary for implementation of BMP, GIF loading functionality (USE_IMAGE_LOADING_IN_X).

PROLOGIO All platforms: necessary for .WXR wxWindows resource-loading functionality (USE_MSW_RESOURCES).

RCPARSER Windows only: necessary for dynamic icon loading (USE_RESOURCE_LOADING_IN_MSW).

Note that if you don't compile in DIB, you could still use wxLoadBitmap in an application and link with dib.lib separately in your application makefile. Similarly, you can use PROLOGIO and RCPARSER independently without them being compiled into wx.lib.

3.1.3. Is CTL3D required?

Here's an excerpt from the wxWindows manual.

It is recommended that CTL3D is used under Windows, since the 3D effects are good-looking and will be standard with Windows 4.0. If you want to use it and don't already have CTL3D installed, copy the files in contrib/ctl3d to appropriate places (ctl3dv2.lib/ctl3d32.lib into your compiler lib directory, ctl3d.h into an include directory, and ctl3dv2.dll into windows/system). You may need to find a compiler-specific version of ctl3dv2.lib or ctl3d32.lib. Define CTL3D to be 1 in wx_setup.h and link your executables with ctl3dv2.lib or ctl3d32.lib.

If both CTL3D and FAFA are set to 1, then all controls except wxButton will use CTL3D and have 3D appearances. wxButton will have the ability to use bitmaps. This is the recommended configuration.

Windows 95 update: dialogs can be marked with the Win95 3D look by specifying the DS_3DLOOK. But this doesn't apply to panels. The WIN32 SDK documentation says that the style WS_EX_CLIENTEDGE can be used for controls, to give them all 3D looks. However, this doesn't appear to work (and causes strange 2-column behaviour in wxListBox). Even marking the executable as Windows 4.0 only gives a wxChoice items a 3D look. So it seems that for now, CTL3D is still required for Windows 95 applications.

3.1.4. I need a drink. Why is compilation so difficult on some platforms?

It's a good question; you may be lucky enough to sail through wxWindows installation without a hitch, or you may exhaust your vocabulary of expletives before you're done compiling the first sample application.

There are a number of possible reasons for things to go wrong:

- Makefiles need to be adjusted (especially make.env) to add include and library paths, and library flags, specific to that OS or compiler.
- I've messed up the distribution. Occasionally I edit a file at the last minute without testing it properly... Normally these problems become apparent quite quickly.
- There's an honest-to-goodness bug in wxWindows. Sorry! but wxWindows is quite complex, and bugs happen. Whether you can classify not coping with a particular setup a bug, I don't know, but there will be occasions when installation reveals a bug. Mostly, though, real bugs are only identified when applications get complex.
- Your compiler has not been installed properly. This is often signalled by missing libraries such as iostream.
- There's a compiler incompatibility. This is extremely rare, since wxWindows uses a very limited subset of C++ syntax, and steers clear of unportable constructs such as templates.
- There's a bug in the compiler. This happens surprisingly often, particularly with GNU C++ where the latest release might have a brand new bug. This can manifest itself as a bizarre link error, or run-time problem such as the message "You must define an instance of wxApp!" (globals haven't been initialized properly by the compiler).
- There's a bug in the OS, such as a lack of certain include files (it happened with some versions of SunOS).
- You're using a compiler and/or OS that no-one's tested wxWindows out on before. If you're really unlucky (and intrepid) you could find yourself doing a 'port' to an environment never before encountered. In fact, the changes involved are usually quite small, and are nearly always centred around wx_utils.cpp and wx_ipc.cpp which make heavy demands on operating system-sensitive areas.

In general, the reason why compiling wxWindows can be more troublesome than other packages is that with conventional application building, you *gradually* use more and more parts of the operating system or GUI toolkit. With wxWindows, because it covers a large 'surface area', you're encountering these possible troublespots all at once when you compile the library.

The good side of all this is that once you *have* ironed out the initial compilation and run-time problems, these particular headaches ought to be minimal from then on. So don't be too

discouraged if installation is initially difficult!

Borland C++ seems to generate the most traffic for installation problems. I'm not exactly sure why this is, since although I don't have Borland C++, various people have contributed tips and makefiles. I suspect that something about the design of Borland C++ makes it difficult to compile a large project without a lot of in-depth knowledge about the compiler options.

3.2. UNIX

3.2.1. Is there a GNU configure script for wxWindows?

Yes, it's in the contrib/wxshlib directory of the distribution, from 1.66 onwards.

3.2.2. Linux issues

3.2.2.1. Segmentation fault on startup

Some versions of gcc are buggy and cause problems with wxWindows and other software.

From Wolfram Gloger:

```
I'm afraid the answer is probably `don't use Slackware' (for C++
development, that is, it may well be a great distribution otherwise).
Slackware has been well known for shipping inconsistent
compiler/library versions.  At this stage, you should really only use
gcc-2.7.2 with libg++-2.7.1.4 (note the trailing `.4' indicating
H.J.'s patchlevel).

A good test would be to compile `#include <stream.h> int main() { cout
<< "Hello\n"; }' and see if that runs.  If it doesn't, you have
obviously no chance to run wxWindows, either.

Regards,
Wolfram.
```

The experience of another Linux user:

```
Hi Julian,

you wrote:
>This is quite a common experience under Linux and it is solvable - I
must
>try to get a coherent story on what the problem and fix is.

you are right, it's very difficult to get a coherent story on what it
depends.
After getting some experience I can definitively say:

1)    At first it is a problem of libg++-2.7.0 -> libg++-2.7.1.3
      Don't use them, they will not work.  SEG-FAULT !!
      You have to use libg++-2.7.1.4 . FOUR !! is very importent,
      thanks Wolfram.
```

```
2)    It is a problem of gcc/g++ 2.7.0 too, exspecially of the linker.
      Maybe it's only the libc.a . I have changed only whole packages.
      It caused a problem with the slider.

3)    libiostream.a came with libgxx-2.7.0 .
      In the package libgxx-2.7.1.4 wasn't any libiostream.
      {{So I couldn't upgrade this library.
        I still have problems, now the choise-box causes failure
        Wolframs little program 'cout' starts with SEG-FAULT.}}
        I just heard it's now itegrated in the new stdc++ and I
        have to delete the old one.

Unfortunately there is no revisions-number and patchlevel-number in the
library-name of libg, ldso-1 and libiostream and I don't know how to
get it.

Point 1) + 2) will help other Linux user if they could find it your
NOTES FOR LINUX USER.

That's all till now.
Thanks a lot for your help and to all the other helpfull people.
Otherwise I didn't search for a solution at this point.

Regards,
Juergen
```

3.2.2.2. I get a link error for strchr

Try adding -lstdc++ to the link flags.

3.2.2.3. Why do the makefiles not work?

You may be using 'pmake': the wxWindows makefiles require you to be using the default GNU make, which has a slightly different syntax (for example, the include statement syntax is different).

3.2.2.4. My binaries are enormous! What can I do?

wxWindows 1.60 improves on 1.50 by the use of GCC pragmas to specify which files are interfaces and which are implementation.

Also, if you compile everything without debugging information, GCC will use dynamic link libraries for X11, XView and some others; this reduces the size of the binary substantially.

You can also create a shared version of the wxWindows library; see contrib/wxshlib for GNU autoconf files for building shared and static libraries, and also *Building shared libraries* (page 10).

3.2.2.5. Building shared libraries on UNIX

See contrib/wxshlib for GNU autoconf files for building a shared version of wxWindows. Here's

another way to do it in Linux:

```
Date: Mon, 10 Feb 1997 21:42:29 +0100
From: Erwin Nijmeijer <E.Nijmeijer@inter.NL.net>
Reply-To: E.Nijmeijer@inter.NL.net
To: julian.smart@ukonline.co.uk
Subject: Shared library & gcc & IOU1

After looking at your homepage I discovered that there is a special
script for creating a shared wxWindows library. To create a shared
version of the library, I used a much easier way :

a) Add -fPIC to CPPFLAGS
b) create the static library as described in the documentation
c) copy the library to a special temporary directory and extract all
   the objects using the archiver :
   ar x libwx_motif.a

d) rebuild the shared library using these objects :
   gcc -shared -o libwx_motif.so *.o

maybe I'm thinking just too simple but it seems to work fine with me !
```

3.2.2.6. How can I reduce wxWindows compilation times on Linux?

A solution from Giovanni Agostino Andrea Giorgi (giorg@dsi.unimi.it).

```
For Linux Users with at least 8Mb of Ram.....
About wxWin 1.63 compilation Speed.
I have found a small solution  for this  problem:

Rules are simple....:)

      1) DO NOT run  X-Windows BEFORE compiling
      2) Include DIRECTLY the ".h" files
      3) Options for gcc:
            -O0 [ -w ]
      4) Try to use more files, to link togther at the end...
With this method I reduced compilation time (and linking, of
course !) of minimal.cpp from over 4' to  about 1':30''.

I think this is good, because I have only 8 Mb....

Thank to all !
Regards
```

3.2.2.7. Why does the Xfree ATI Mach32 server hang when drawing graphics?

(This FAQ is probably obsolete by now).

Harri Pasanen has discovered a bug in the Mach32 server. It hangs if using pens with wxDOT linestyle, and width zero.

This has been reported to the Xfree developers.

3.2.3. Solaris 2.x issues

3.2.3.1. I get some warnings and link errors. What gives?

You need to:

- Compile with -DSVR4. Add this to OPTIONS line in each makefile.unx.
- Add the following to LDFLAGS: -lgen -ldl -lsocket -lnsl

Note that the libgen.a lives in /usr/ccs/lib, if you have installed the programming tools option.

Version 4.0 of Sun C++ is apparently more pedantic than older versions, and requires the use of CC -migration to help with the necessary changes. You may need to include the file strings.h where the file string.h is included, with CC, e.g. in wb_utils.cpp.

Someone reported that link errors on a SPARCStation were cured by adding -lucb and -I/usr/ucbinclude/sys.

For dynamic linking under Solaris 2.3, the following changes are required:

- In wxinstal, add:

```
export OPTIONS
OPTIONS=-fPIC
```

- in src/x/makefile.unx, add:

```
WXLIB = $(WXDIR)/lib/libwx$(GUISUFFIX).so.0

$(WXLIB): $(OBJECTS) $(BASEOBJECTS)
            ld -G -o $(WXLIB) $(OBJECTS) $(BASEOBJECTS)
```

  where the ld line replaces the ar+ranlib command.

Here are my (JACS) own experiences. As of 25th May 1995, I finally got a clean compilation under Solaris (using XView), though I needed to change a lot of files, e.g. in wxXPM and wxImage. The following is a script I wrote to save editing make.env, for Solaris compilation. It shows the kinds of settings required.

I think for some environments you may need to add -L/usr/ccs/lib -lgen to the COMPLIBS line.

The changes required to compile under Solaris will be in version 1.62 beta (b). New versions of Solaris and the SunPro compiler may break all this, of course.

```
#!/bin/sh
```

```
# makeunix
# Invokes makefile with specific XLIB and XINCLUDE settings,
# IFF your version of make can take the -e flag
# (environment variables take precedence.)
export XINCLUDE
export XLIB
export CC
export CCC
export CCLEX
export DEBUG
export WARN
export RANLIB
export COMPLIBS
export OPTIONS
CC=CC
CCC=cc
CCLEX=cc
OPTIONS=-DSVR4
COMPLIBS='-ldl -lsocket -lnsl'
XINCLUDE=-I/usr/openwin/include
XLIB='-L/usr/local/X11/lib -L/usr/openwin/lib'
DEBUG=
WARN=
RANLIB=echo
make -f makefile.unx -e $@
```

3.2.4. Compiling on OSF/1

Here's how.

```
From: Asociacion Fisica Universidad <afu2@eucmos.sim.ucm.es>
Subject: Ported wxWin 1.60 to OSF1
To: J.Smart@ed.ac.uk

        Hi!

        I've been playing around with wxWin (great package!) and I've
make it to compile under OSF/1 with motif. I send you the modified
make.env, just in case.

        There is a minor change in a file I can't remember which is,
but
is in someplace in which it makes a wait and you say it's bad, that it
has to be remade. There's a conditional compilation there, and where we
can fidn #if !defined(SVR4) && .. etc, just include a !defined(OSF1).
It
shoudl work all right.

        Well, here's the make.env.osf1. If you have any comments, let
me know!

# make.env

# slightly touched by Iniaky Perez Gonzalez (afu2@fis.ucm.es,
2:341/5.31)
```

```
# to work fine under OSF/1

# Common makefile settings for wxWindows programs
# This file is included by all the other makefiles, thus changes
# made here take effect everywhere (except where overriden).
#
# An alternative to editing this file is to create a shell script
# to export specific variables, and call make with the -e switch
# to override makefile variables. See wx/install/install.txt.
# And you can override specific variables on the make command line,
e.g.
#
# make -f makefile.unix DEBUG=''
#

########################## Compiler ###################################

# C++ compiler
#CC = gcc-2.1
CC = cxx

# C compiler for pure C programs
# Typical: CC=g++ , CCC=gcc
#          CC=cl386 /Tp, CCC=cl386
#
# (Used only for XView, file sb_scrol.c)
#
CCC = cc

# Compiler used for LEX generated C
CCLEX=$(CCC)

########################## Compiler flags #############################

# Miscellaneous compiler options
# May need to add -D_HPUX_SOURCE_ for HPUX
# Solaris: add -DSVR4
OPTIONS= -Dosf1 -DOSF1 -D__OSF1

# Debugging information
#DEBUG = -g
DEBUG =

# Warnings
WARN =

# Which GUI, -Dwx_xview or -Dwx_motif (don't change this)
GUI = -Dwx_motif

# Optimisation
# OPT = -O
OPT =

# Options for ar archiver
# AROPTIONS = crs # For IRIX. Also, comment out ranlib line.
AROPTIONS = sruv
```

```
# Compiler libraries: defaults to GCC libraries
# Sun with Sun CC: -lc
# Solaris: -lgen -ldl -lsocket -lnsl
#   and/or possibly -lucb, whatever that is...
# SGI:     -lPW
COMPLIBS=-lc -lm -lcxx

# Compiler or system-specific include paths
# E.g. some SPARCStations need
# -I/usr/ucbinclude/sys
COMPPATHS=-I/usr/include/cxx

# HP-specific compiler library: an AIAI convenience
HPCOMPLIBS=

# LDLIBS for specific GUIs
MOTIFLDLIBS = -lwx_motif -lXm -lXt -lX11 -lm $(COMPLIBS)
XVIEWLDLIBS = -lwx_ol -lxview -lolgx -lX11 -lm $(COMPLIBS)
HPLDLIBS=-lwx_hp -lXm -lXt -lX11 -lm

# Default LDLIBS for XView (don't change this)
LDLIBS = $(XVIEWLDLIBS) -lbsd

# _ol or _motif (don't need to change, the makefiles will take
# care of it if you use motif/hp/xview targets)
GUISUFFIX=_motif

########################## Directories #################################

# Replace X include/lib directories with your own
INCLUDE=-I/usr/include -I/usr/include/X11 -I/usr/include/Xm
LIB=-L/usr/local/X11/lib -L/usr/lib/Xm
#XINCLUDE=-I/aiai/packages/motif1.2.1/motif/include -
I/aiai/packages/X.V11R5/inc
lude
#XLIB=-L/aiai/packages/motif1.2.1/motif/sun4/lib -
L/aiai/packages/X.V11R5/lib

# A convenience, for HP compilation
HPXINCLUDE=-I/usr/include/Motif1.2 -I/usr/include/X11R5
HPXLIB=-L/usr/lib/Motif1.2 -L/usr/lib/X11R5

# Shouldn't need to change these...
WXINC = $(WXDIR)/include/x
WXBASEINC = $(WXDIR)/include/base
WXLIB = $(WXDIR)/lib/libwx$(GUISUFFIX).a
INC = -I$(WXBASEINC) -I$(WXINC) $(COMPPATHS)

# Directory for object files (don't change)
OBJDIR = objects$(GUISUFFIX)

# You shouldn't need to change these...
CPPFLAGS = $(XINCLUDE) $(INC) $(OPTIONS) $(GUI) $(DEBUG) $(WARN) $(OPT)
CFLAGS = $(XINCLUDE) $(INC) $(OPTIONS) $(GUI) $(DEBUG) $(WARN) $(OPT)
LDFLAGS =  $(XLIB) -L$(WXDIR)/lib

# Extra patch link for XView
```

```
XVIEW_LINK = $(WXDIR)/src/x/objects_ol/sb_scrol.o
```

Also:

```
Hi,

just one word on how to compile Wx166b on DEC/OSF

The DEC cxx compiler does'nt understand the new .cpp extensions as C
plus plus
source files.

Solution
---------

modify the make.env file to add -x cxx to the compiler options

->   OPTIONS = -Dosf1 -DOSF1 -D__OSF1 -x cxx

the -x cxx argument forces to compile any source to C plus plus

(see the FAQ for a complete list of other changes to introduce in the
make.env file)
Facultes Universitaires Catholiques de Mons (F.U.Ca.M.)
Bart JOURQUIN
Departement "Informatique et Gestion Quantitative"
Groupe "Transport et Mobilite"
Chaussee de Binche, 151a
7000 Mons (Belgique)
Tel: (32) 65 32.32.93
Fax: (32) 65 31.56.91
E-mail: jourquin@message.fucam.ac.be
```

3.2.5. Compiling on HP kit

Here's how.

```
Date: Fri, 21 Apr 1995 09:03:32 -0600
From: Bruce Lee <lee@abraham.et.byu.edu>
Apparently-To: J.Smart@ed.ac.uk
Status: REO

Julian,

Thank you for the suggestions concerning the wxEntry problem I was
having.  I
changed main.c to a c++ file and commented out the extern C wxEntry in
wx_main.c
c
and all was well.  If you or the wxWindows users are interested the
following
are the changes I had to make to get wx161 to compile on an HP 7xx/8xx
machine
using HP's C++ compiler:
```

```
        * Add the compiler flag +a1 to the options field in
src/make.env
          This tells the compiler to be ASNI strict.

        * You _must_ use flex _and_ bison to compile y_tab.c in the
prologio
          stuff.  Also I found gcc works best to build y_tab.o.

        * In contrib/xmgauge/gauge.c, change #ifdef 0 to #if 0

NOTE: The standard c compiler cc on the HP will warn you that +a1 is an
invalid
      option when building non-C++ files.  Gcc will bomb if that option
is used
      when building y_tab.c.  You can hack the makefile or create a new
macro
      to provide the proper options.

Thank you,

Bruce
```

3.2.6. Compiling Sun dynamic libraries

(See also contrib/wxshlib).

From Frank Brueggemann:

```
From: Frank Brueggemann <fjb@newton.fb5.uni-siegen.de>
Date: Fri, 17 Mar 95 09:07:35 +0100
To: wxwin-users@aiai.ed.ac.uk
Subject: Re: Dynamic libraries on Sparc question from Keith

Yesterday my colleague Dominik and I succeded in compiling
wxwin as a dynamic library with gnu 2.6.3 and libg++ 2.6.2 for
SunOS 4.1.3. and openwin 3.0.

The behaviour mentioned is a normal result of the dynamic library
system,
but it not so obvious at the first moment. It took some time for us
to figure out what to do. SUN distinguishes between funtional shared
libraries (so called .so files)
and libraries that exports initialized data (so called .sa files).

If you wish using the wxwin library as a dynamic library you have to
create a libwx_??.sa file too. This is necessary because the files
    src/base/objects_ol/wb_main.o \
    src/base/objects_ol/wb_obj.o \
    src/base/objects_ol/wb_types.o \
    src/x/objects_ol/wx_main.o
contain such global data. Thus you have to use a command like
ar rv libwx_ol.sa.0.0 \
    src/base/objects_ol/wb_main.o \
    src/base/objects_ol/wb_obj.o \
    src/base/objects_ol/wb_types.o \
```

```
    src/x/objects_ol/wx_main.o

to build this library in addition to the libwx_ol.so.0.0.

I think the compilation is very similar on the SOLARIS OS.
If you need further details please reply.
```

From Mikhail Tcheznychev:

```
Sme words about dynamic library for Solaris 2.4 with
gcc-2.7.0

    In make.env, options:

        OPTIONS=-DSVR4 -fPIC

    In src/x/makefile.unx :

$(WXLIB): $(BASEOBJECTS) $(OBJECTS) $(EXTRAOBJS)
      gcc -G -o $(WXLIB)  -h libwx_ol.so.0 $(EXTRAOBJS) $(OBJECTS)
$(BASEOBJECTS)
\end{verbaim}


\subsection{I get link errors for wxEntry, LexFromFile etc.}

Sometimes you might get some or all of these symbols undefined when
linking a sample:

\begin{verbatim}
  wxEntry(int, char**)
  LexFromFile
  PROIO_yyparse
  LexFromString
```

For PrologIO, try setting CCLEX in make.env to use the C compiler, not the C++ compiler. With the wxEntry problem, if all else fails, change wxEntry in wx_main.cpp to main, and don't link with main.o.

3.2.7. I get a link error under SunOS: the symbol XtShellStrings is resolved.

Tako Schotanus (sst@bouw.tno.nl) writes:

```
I was finally able to solve it by adding the following define
to "make.env" :

  -DXTSTRINGDEFINES

This has the effect that whenever there's a reference in the
sourcecode to XtN..... (XtNiconName for example) a #define with
the proper string will be used instead of a global array containing
the names.

System: SunOS 4.1.3
libXt:  4.10
```

```
gcc:    2.5.8
```

Another cause may be having multiple versions of libraries (such as the Motif library) in your path.

3.2.8. I get a libXmu link or run-time error.

Put -lXmu in your LDLIBS. This library is used by the ComboBox widget; if you can't get rid of the error, try setting USE_COMBOBOX to 0 in wx_setup.h and recompiling wxWindows and the application.

3.2.9. How do I link applications statically with X and Motif libraries?

Sometimes it's desirable to link an application statically, if the recipient of the executable may not have the appropriate dynamic link libraries. The tradeoff is larger executable sizes.

Here are some responses to a query I put to wxwin-users.

```
If you're doing this on Solaris, try at the end of your $(CC) command

 -Wl,-Bstatic -lXm -lXt -lXmu -lX11 -Wl,-Bdynamic -lgen -lnsl -lsocket

for SunOS - just do -Wl,-Bstatic

for HPUX or AIX I wouldn't worry about it and I'm not sure about
other platforms - Linux might be a problem ...

Hope this helps.

-jb

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

gcc -static
works for me.

You can also just put the .a files on the command line if you only want
to link some libs statically, eg
gcc -o prog ob1.o ob2.o /usr/X11/lib/libXm.a  \
      /usr/X11/lib/libXt.a /usr/X11/lib/libX11.a -lm

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

Just give the  -Bstatic flag to the linker command line, after the -l
flags.

Rajive

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

It's not hard.  If you are using gcc, then supply -static before the
list of
link libraries -- i.e. the set of -l parameters (such as -lXm).  If you
are
using Sun compilers use -Bstatic.  Just add this flag to the LDFLAGS
```

```
parameter in make or imake.  You know you've succeeded when the ldd
command
for the created executable returns `statically linked.'

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

Hi,

> Just give the  -Bstatic flag to the linker command line, after the -l
flags.

This is in SunOS, presumably.  On Linux/ELF, it's -static, and
_before_ all -l flags (since you can turn on/off dynamic linking
on a per-library basis).  It's a pity not all Un*x linkers are
the same...

Regards,

Wolfram.
```

3.2.10. What to do if GCC gives non-wxWindows link errors

Here's a success story from David Starr (dave@doom.sbi.com), who got the following link errors using GCC 2.5.5. He upgraded to 2.6.2.

```
ld: Undefined symbol
   _mktemp__FPc
   ___9streambufRC9streambuf

Blymn,

I am writing to thank you for your help in carrying me through
to a successful build, at last, of the wxwin 1.63 release.

Especially since the outcome was positive, I thought I would
share with you my interpretations and the specific results
of your recommendations.

You wrote:

 >> One thing that did give me these sorts of problems was a faulty
libg++
 >> library.  The problem is that Sun's make handles the output of
shell
 >> commands on command lines differently to that of GNU make (Suns
dumps
 >> the output as one big argument on the command line while gmake uses
 >> the IFS shell env variable to split the output into arguments).
What
 >> it boils down to is that if you did not compile libg++ using gmake
 >> then the library is seriously broken.  If you try running nm -p on
 >> libg++.a and get a fairly short list of things defined then the
 >> library is broken, if the list goes on and on then look elsewhere
for
 >> the problem.
```

I discounted the suggestion of the faulty libg++ at first because
the indication from running the nm command was that there were lots of
references. However, as time went along, I became more convinced that
this was the problem, and it was further compounded by libg++ and gcc
being out of sync. I had been running gcc 2.5.5.

I went to my sa and asked him to upgrade my gcc. Being the
enthusiastic sort, he went about upgrading /usr/local/bin/gcc which
clobbered the 2.5.5 compiler in use by about 10 members of our group.
After that was restored, and everyone settled down, I continued.

Our sa then gave me a separate gcc2.6.3. Then, also thanks to you,
I got by ftp libg++-2.6.2 from prep.ai.mit.edu (no one here seemed
to know the exact site name), and was able to compile
it successfully. By this time I had forgetton your suggestion to use
gmake, but the result still worked.

 >> Uhh the symbols given are the c++ mangled names. The GNU binutils
has
 >> a name demangler called c++filt. The names you are having problems
 >> demangle to:
 >>
 >> mktemp(char *)
 >> streambuf::streambuf(streambuf const &)
 >>
 >> You could try defining these two and see what happens but I doubt
if
 >> you will get far. I still reckon that your g++ lib is stuffed.

However, I still was getting
  ld: Undefined symbol
     _mktemp__FPc

So, thanks to your post, I had learned about c++filt, and was able to
define mktemp(char*) in my source, resulting in successfully linked
load modules.

3.2.11. Why does the XView or Motif file selector crash?

> Hi, I compiled wxwin 1.61 beta with gcc 2.6.3 on a Sun sparc.
Everything work
s
> except in the hello demo when I try to open the file selector the
program
> crashes. Any clue on this is appreciated. (The same problem does not
> occur using gcc 2.4.5. Also I use motif 1.2.)

I had the same problem. I believe it is *not* the fault of wxWindows.
Instead, the problem appears to lie in the librx (regular expression)
code that is distributed with the most recent version of libg++
(2.6.2?) -- at least, that's where it's crashing.

What I did was to delete all the librx code and all references to it
in the libg++ makefiles, then recompile libg++. (Just for safety's

```
sake, I recompiled the wx library as well; I wasn't sure whether it
would have picked up any of the librx code.)  When you relink your
application, the regular expression code in the system libraries will
be used instead.  File selectors now work fine for me.

You may be able to achieve the same effect by making sure that libg++
is the absolute *last* library searched by your compiler (after the
system libraries, in particular), but I haven't tried this.

-------------------------+---------------------------------------
   //  Scott Maxwell:    |
\\//      maxwell@       | ``Unlike most of you, I am not a nut.''
 XX natasha.jpl.nasa.gov |     -- Homer Simpson
```

Here's another solution that doesn't involve recompiling libg++:

```
------------------
wxWindow 1.62e
HP-UX arcturus A.09.04 U 9000/887
gcc version 2.6.3
------------------

to extract rx.o from libg++:

ar d libg++.a rx.o

Warning !
In case you created a copy of libg++.a, let say libwx_g++.a, whithout
rx.o (cp libg++.a libwx_g++.a; ar d libwx_g++.a rx.o), but let the
original version of lig++.a in place not to disturb other users of
the library; be sure to call the compiler from your makefiles as "gcc",
because "g++" is an alias which automatically generates a reference to
libg++ (-lg++). Thus even if you would mention -lwx_g++ in you
makefiles,
your changes might seem not to be operative.
```

And yet another, even simpler solution:

```
By the way, I want you to know that libg++ should go before
libXm.  The way you have it set up in makefile.unx, make.env,
the file - load core dumps in XmCreateFileSelectionDialog.
The reason is that libg++ has a re_create function which
apparently is the same name as a motif function used by
XmCreateFileSelectionDialog.

David Starr
```

3.2.11.1. ULTRIX compilation

- Link the iostream library or you will get link errors.
- Define NEED_STRDUP to 1 in controb/wxxpm/libxpm/libxpm.34b/lib/xpm34p.h.
- Apparently some of the extra libraries (prologio, image etc.) need to be specified in the sample or application makefile to avoid link errors: I don't know why this should be since the objects should be linked into the main libwx_motif.a file.

3.2.12. Under AIX, wxTheApp does not initialize properly and causes a wxWindows error message.

After getting wxWindows 1.61 (b) to compiler under AIX, Dirk Eller writes

```
The major problem was the initialisation problem of the wxApp-object
also reported in install/install.txt.
The fix:
  #ifdef __aix
  extern wxApp *wxTheApp=1;
  #endif
doesnt work on my system. It looks also very bad ?

After a little testing I figured out that wxApp isn't initialized at
the time
when main() (->wxEntry) is called.
The fix is to move the call of main() to the file where the global
object
is supposed to be created. (e.g. hello.cpp)

The reason is that (stroustroup) c++ compiler does not HAVE to create a
global
object before main, but only before any functions are called in the
file that
the object is in. (thanks Eugene)
The fix is not very elegant, but it works.
```

3.2.13. When deleting a frame or dialog box, the program crashes.

On some systems, you should not use the delete operator to delete frames or dialog boxes. Use wxPostDelete instead, to get the object deleted when X has finished processing all other messages.

Note for version 1.66 and later: wxWindow::Close has been introduced for delayed deletion of frames and dialogs. Please use this instead of wxPostDelete.

3.2.14. How can I compile PROLOGIO successfully under UNIX?

Check the CCLEX variable in src/make.env; set it to use a bog-standard (or GNU) C compiler for compiling LEX-generated files.

Using FLEX instead of LEX sometimes helps, too.

Most of the warnings when compiling PROLOGIO are spurious; however, there may be an error buried inside the warnings. If so, you may need to change a prototype in a generated .c file to get it compiled. Hopefully this type of error is getting much rarer now.

3.2.15. How can I compile wxXPM successfully under UNIX?

Check the XPMCOMPILER variable in contrib/wxxpm/makefile.unx; set it to use a bog-standard (or GNU) C compiler.

3.2.16. I get libXmu.so.4 (or similar) not found on linking.

This library is currently only needed for the ComboBox implementation. If this is causing trouble, switch off ComboBox compilation in wx_setup.h and src/x/makefile.unx, and recompile.

3.2.17. Why do I have compilation problems compiling wb_list.cpp on Solaris?

If you get this kind of message:

```
wb_list.cpp: In method `wxList::wxList(wxObject * ...)':
wb_list.cpp:178: `__builtin_va_alist' undeclared (first use this
function)
wb_list.cpp:178: (Each undeclared identifier is reported only once
wb_list.cpp:178: for each function it appears in.)
wb_list.cpp:186: warning: implicit declaration of function `int
__builtin_va_arg_incr(...)'
```

try editing the makefile and setting your OPTIONS (XView/Motif) or FLAGS (Xt) as follows:

```
FLAGS = -O2 -D__EXTENSIONS__ -Dsparc -Dlint -DSVR4 -Wall -Dwx_xt # (or
-Dwx_motif for Motif)
```

3.2.18. Why do I get an undefined virtual table error?

If you get something like this:

```
 ld: Undefined symbol
    wxView virtual table
    wxCommand virtual table
    wxFileHistory virtual table
    ...
```

when compiling with g++, it could be that the relevant file (here, src/base/wx_doc.cpp) is in DOS CR/LF format. Remove the extra characters with a dos2unix utility, or by zipping up the file and unzipping with the -a option, or by using a text editor, and recompile.

3.2.19. Unresolved references using Linux and SWiM Motif 2.0

If you get unresolved references such as:

```
Shell.o(.text+0x2ba0): undefined reference to `SmcModifyCallbacks'
```

you may need to add /usr/X11R6/lib to your XLIB variable in src/make.env.

3.2.20. Duplicate symbol error when compiling with xlC on AIX 4.1.4

It's not necessary to use "-lCns -lbsd" when linking wx applications using xlC (AIX 4.1.4).

```
make.env:

# AIX: -lCns -lbsd
#COMPLIBS=-lCns -lbsd
COMPLIBS=
```

3.3. VMS

These are Stefan Hammes' notes for compiling wxWindows on VMS. They were written for wxWindows 1.61b but most points should hold true for later versions.

```
This port of wxWindows 1.61b is for the DEC C++ compiler on VMS
for ALPHA and VAX. I'm using an ALPHA, so I cannot guarantee for
no problems on a VAX, but if there are problems, they will be
minor ones (mostly with include files etc.).

This port is not a complete one, but all graphical features work.
Timer works. Most other things work.
The toolbar works fine. All(!) samples (except the IPC one) work.
Things which do not work: IPC (not yet because I don't have sockets),
PROLOGIO and most other utilities (I'm working on this) and the
contrib stuff.

Most problems occur because of the directory structure and
filenaming of VMS. Beside this, some system dependent headerfiles
of UNIX are not present under VMS.

The directory structure is the same as under UNIX.

Be warned: The DEC C++ compiler is very slow. On a VAXstation 3100 the
compilation time is about 24 hours (!!!). It also needs much much
memory (surely you have to raise your pagefile size ;-).

Files
-----
Get a copy of wxWindows and put the directory hierarchy on
your disk. Copy the file \include\base\wx_setup.vms to
\include\base\wx_setup.h.
Then you should have a ready version of the source code.

Environment
-----------
Under all circumstances you should make the following definition
in your 'login.com' file:

$ make ==
"mms/descrip=makefile.vms/macro=(ALPHA=1,WXDIR=[hammes.wxw161)"

Without this, nothing works :-)
Instead of ALPHA=1 you can use VAX=1 if you are on a VAX.
The WXDIR should point to the directory of wxWindows and you MUST omit
the
trailing ']' !!! Replace the string 'hammes.wxw161' with the correct
one
for your system.
```

```
In $(WXDIR).src] we have the two files 'makevms.env' and 'motif.opt'.
They define options and locations of directories and libraries.
Edit them for your system. For the first try no editing is necessary.
'makevms.env' will be included in the makefiles so you need to
define the things only once.

!!!! IMPORTANT !!!!
Now you should delete the file

'$(WXDIR).include.base]wxstring.h'

and copy the file

'$(WXDIR).include.base]wx_setup.vms' to
'$(WXDIR).include.base]wx_setup.h'
!!!! IMPORTANT !!!!

Makefiles
---------
I have included makefiles with the name 'makefile.vms' in several
directories.
If you have defined the above symbol 'make' and have 'mms' installed on
your system you need only to type 'make' as in UNIX.

N.b.: Instead of having makefiles in [.src.x] AND [.src.base] we only
have
      one makefile in [.src.x] which makes the whole library.

You can look at the makefiles and build similar ones for other
libraries and
for executables.

Compiling
---------
If everything goes right, you have only to type 'make' in the
top wxWindows directory and then go to sleep or something else.
This global make-command will build the wxWindows library,
the toolbar library, the wxstring library and all sample programs.

As stated above this will take a lot of time (1-2 days :-() for
compiling all components). On an ALPHA it will be somewhat
faster.

VMS port was done by Stefan Hammes (stefan.hammes@urz.uni-
heidelberg.de).
In the source code I have marked my changes and additions with 'steve'
for the old changes and 'steve161' for the newest changes.

If you have problems, please send an E-mail to me. I wish you success,

Stefan Hammes
```

3.4. MS Windows

3.4.1. What's the best compiler to use for Windows programming with

wxWindows?

There will be a variety of views on this, but here's my view (Julian Smart).

Borland C++ 4.x generates a lot of mailing list traffic with people experiencing a bewildering variety of problems. The size and scope of wxWindows exacerbates any problems with a Windows compiler, and Borland is no exception (see later sections for more details). The debugger is particularly bad, but then this is a general problem with Windows compilers. Borland has the major advantages of wide use and 32-bit WIN32S, but I would not recommend it over Visual C++ 1.x if you are starting out with a choice and do not need 32-bit compilation.

Visual C7 should work with wxWindows but I'd recommend upgrading.

Visual C++ 1.5 is what I use for wxWindows and therefore the makefiles are the most developed, and there will be least trouble in using wxWindows with Visual C++ 1.5. The CodeView debugger, while boring, is at least reliable and functional unlike many of the others. However, for 32-bit compilation you'll need a separate compiler (e.g. VC++ 2.x which comes with VC++ 1.5 on the same CD-ROM). I haven't used VC++ 2.x but it sounds at least as reliable as 1.5.

Watcom C++ 10.0 is famous for its generated code speed and ability to compile in 32-bit mode. Users have spent some time and trouble making Watcom C++ and wxWindows see eye-to-eye, although there are still a few wrinkles to be resolved. However 16-bit wxWindows compilation with Watcom is a no-no in my experience. The text-mode debugger seems OK but flashes like mad between text screen and Windows screen so CodeView looks like heaven in comparison. It's possible that your video card, if not your eyes, would give up the ghost after a lot of this kind od switching. The Windows-hosted debugger is itself to buggy to use, unfortunately. Watcom boasts a wide range of targets for the compiler, including 16-bit Windows, 32-bit standard Windows API, 32-bit WIN32 and WIN32s API, and even OS/2 if you purchase more kit from IBM for Presentation Manager programming. Unfortunately you won't be able to debug WIN32s programs under Windows 3.1: you'll need Windows 95 or NT for that. Watcom compile speed is effectively very slow because the precompiled header criteria are too strict to be practical; but this is partially compensated for by quick link times. So if you're doing small changes to the source and lots of linking, Watcom is much better than Visual C++.

Symantec C++ I have no experience of, though some people use it with wxWindows. It has a famously nice IDE, but so far I've avoided IDEs because of their lack of flexibility compared with makefiles, and they have a habit of multitasking badly.

DJGPP (a port of GNU G++) is not a contender (see section on this below).

In summary: if you want to be reasonably sure of reliability and compatibility with future versions of Windows and wxWindows, I would recommend going for Microsoft's VC++ 2.x (a 32-bit compiler with 16-bit VC++ 1.5 included). Not because I enjoy swelling Microsoft's profits, but I believe this is the most reliable and hassle-free solution. If your application doesn't need 32-bits or fancy controls, then you could do worse than compile 16-bit applications using VC++ 1.5 ensuring that they'll run under NT/Windows 95; and then make the switch to 32-bit compilation when you know most people can run your 32-bit applications.

With the hassles implicit in GUI programming, the last thing you need is an unreliable compiler, so with compiler prices dropping, I'd recommend that you treat yourself to a decent compiler.

3.4.2. Can I use an Integrated Development Environment?

It is possible to use an existing IDE for writing code, but the project file settings are tricky and

often the IDEs do not allow the flexibility that wxWindows requires, with its little libraries tucked away in odd places. By adding enough switches, it should be possible.

In particular, some IDEs (such as Microsoft's) do not like C++ source files to have .cpp extensions, so renaming of .cpp files is necessary.

3.4.3. How can I use wxWindows with Borland C++?

Here are some tips for using Borland C++ with wxWindows.

3.4.3.1. To create new IDE project files

1. The IDE must recognize .cpp files to edit and compile wxWindows source. You must add .cpp to the extension lists in the Options/Tools/CppCompile, Options/Tools/EditText and Options/Environment/Syntax Highlighting menus.

2. Create a new project file with the following settings:

    - Platform: Win16 or Win32 (always the same)
    - Standard libs: No OWL, no Class Lib (unportables)
    - Advanced Ops/No source node

3. If the source files already exist, just type Ins on the target name (*.exe or *.lib) of the project window. The selector file dialog must appear. Then select all the relevant *.cpp files, and the *.rc and *.def files.

    If you are not sure about which files must be included, look into a makefile, if it exists, for the sources. Don't worry about libraries, BC4 has its own windows libraries.

3.4.3.2. To use old *.prj project files

The IDE can read version 3.1 *.prj files, and then convert them to the new format. You must repeat item 1 (above) each time you read a *.prj file, and sometimes you'll have to select the .cpp files in the project window, and drop them on the target file (.exe or .lib), in order to obtain little bullets in left of the project window (ie. to make the IDE recognize the files). If there are no many .cpp files, I recomend to create a new *.ide file.

3.4.3.3. To use makefiles

Now the wxWindows distribution includes Borland makefiles. The file "makefile.bcc" from `samples\minimal` directory could be used as a template.

3.4.3.4. To compile wxWindows projects

Build `wx.lib` with the supplied makefiles. If you want to use the IDE for your new applications:

1. Set the proper directories. In the menu Options/Project/Directories/Include, add

`"(wxdir)\include\base;(wxdir)\include\msw"`, where (wxdir) must be changed for the wxWindows base directory. Avoid absolute paths in your code, if you want it to be portable.

2. In Options/Messages/Potential C++ errors menu, uncheck "function f1 hides virtual function f2", to reduce the amount of message warnings during compiling.

3. A typical Windows application (16bits mode) can't have a data segment larger than 64k. To avoid *data segment overflow* (page 32) and linking problems, set in Options/Project/16-bit Compiler/Memory Model:

    - Memory model: large
    - Asume SS = DS: never
    - Far virtual tables: on
    - Automatic far data: on
    - Far data threshold: 512

Example files for minimal example:

```
 minimal[.exe]
     -minimal.cpp
     -minimal.rc
     -minimal.def
     -wx.lib
```

3.4.3.5. Borland C++ complains about a missing 1.cpp file.

Copy wx_panel.cpp, wx_utils.cpp, wx_dialg.cpp, wx_timer.cpp and wx_clipb.cpp to corresponding .cpp files. You need not change the names in the makefile. BCC will compile the .cpp modules.

Hopefully this workaround will be made redundant at some point in the future.

3.4.3.6. Using GNU wxString/wxRegex with Borland C++

```
From: leif@danmos.dk
Date: Wed, 2 Apr 97 02:36 WST
To: wxwin-users@babbage.eng.nene.ac.uk
Subject: Borland C++ 5.0 and wxString problem

    Hi Pasquale and others,

On Fri, 21 Mar 1997 Pasquale Foggia wrote:

> leif@danmos.dk writes:
>     [...]
> > Borland C++/5.0. I am using wxString and wxRegex from 'contrib',
which
> > works fine in Windows 3.11 and Linux. Now I get a memory access
violation
```

```
> > using wxString::Index (wxRegex (" \n\t]+", 0). I have tracked it
down to
>     [...]
> >     I have now tracked the problem down to being a question of
signed/
> > unsigned char's. I am using an .ide to generate all the libraries,
and
> > I have here marked 'Borland extentions' in the 'Compiler|Source
code'
> > option. This seems *not* to set the define __STDC__ which is needed
in
> > wxregex.cpp module. I tried to mark 'ANSI' instead, which made
quite
> > a bit of difference. Now wxRegex worked :-). BUT, if I compile the
> > whole library with this 'ANSI' marked, I get an error in winreg.h
:-(.
> >
> >     I also tried to compile all of the libraries using the
makefile.b32,
> > which generated a library having the first problem (wxRegex not
working).
> >
>     [...]
>
> The Borland C (correctly?) doesn't define __STDC__ when you enable
the
> Borland extensions, because in this mode there are non-ANSI keywords
like
> far (if you are compiling a 16 bit app) or _export that may be non
compatible
> with a valid ANSI C program. The problem is that such keywords are
needed
> to compile <windows.h>, so you must set the Borland extensions
checkbox
> in your project. The solution to your problem is simply putting
>
> #if !defined(__STDC__) && defined(__BORLANDC__)
> #define __STDC__ 1
> #endif
>
> at the beginning of every file that may need it.

    This was really a great help. I had a hard time, however, to track
down which files needed this, until I just tried to put '__STDC__=1' in
Options|Project|Compiler|Defines in the .ide as a general define for
compiling the whole library (it corresponds to the makefile.b32 file).
Then everything worked (and still does :-).

    Maybe it was an idea to put this '#define __STDC__ 1' thing into
the standard makefile.b32 !?


    Thanks,

    Leif Jensen (leif@danmos.dk)
```

3.4.3.7. Miscellaneous tips

If Borland crashes or runs out of memory during a compile, you may need to configure your DPMI swap file: see P. 21 of the Borland 4.0 User Guide. This is particularly necessary if you only have 8MB of physical memory.

For precompiled headers to work properly on the wxWindows library, you will need to uncomment the first

```
// #include "wx.h"
```

line in each file in src/base. Yes, this *does* look like some files will be included three times: but they won't really. The line has to be before any pragmas of ifdefs, or the precompilation doesn't work.

If you don't recompile wx.lib often, this editing is not necessary.

Contributed by Aguilar Sierra Alejandro-CCA <asierra@servidor.unam.mx>, Oliver Niedung <niedung@cip.med-informatik.uni-hildesheim.de> and Andrew Davison <adavison@ozemail.com.au>

3.4.4. How can I use wxWindows with Turbo C++ 3.1?

Here are the settings you need to make in Turbo C++ 3.1:

```
Compiler
 Code Generation:
    Model = Large
    Assume SS=DS = Never
    Defines = wx_msw
    Options = Treate enums as int
              Dup Strings merged
 Advanced Code Generation
    Options = Generate underbars
    Auto Far Data  - Far data threshhold = 4
Make
  Run Librarian
C++ Options
  C++ Always
Link Libraries
  Container Class = none
  Object Windows Lib = none
  Standard Run Time Lib = Static
```

You will probably need to turn off many components in wx_setup.h.

3.4.5. How can I use wxWindows with Borland C++ 3.1?

The simple answer is probably "don't try, get a more recent compiler", but if you really need to, here are some tips to at least get it compiled, albeit with a GPF when running the samples.

From Daniel Schmitz (dschmitz@ch.hp.com):

```
Well, just in case you have users that still have Borland C++ 3.1 AND
windows 3.x...

src/makefile.bcc
    Change the Borland version to 3.1.  I'm not sure this has any
    affect.

wb_timer.cpp
    bcc 3.1 does NOT put underscores before timezone or the other
    variable.

msw/makefile.bcc
    make kept screwing up when recusive making wxstring.  It turns
    out that you pass the command-line option: DEBUG="0"
    make kept complaining that it couldn't find the target 0.  So I
    modified this makefile to pass: -DDEBUG="0"
    and it went through wxstring ok.  I double checked the command
    line options being passed to make.  They were valid, so there must
    be a bug in this version of make.

wx_setup.h
    There aren't any libraries that provide the SQL functions.  Set
    the flag USE_ODBC to 0.

wx.lib
    My stupid stupid stupid... copy of tlib absolutely refused to
    create a library larger than 1MB.  It didn't matter what values I
    gave to the -P option.  I could build a 990K library, then try to
    add one 20K object and it would instantly complain: Out of Memory
    I have 14MB free for tlib to add that extra object file so I don't
    think it's a ram limitation.  I went around this by splitting
    wx.lib into 5 libraries, wx1.lib...wx5.lib.  Before linking sample
    programs, I modified their makefiles to link against all of the wx
    libraries I built.

erase f1 f2 f3...
    The makefile.bcc files in the following contrib directories gave
    multiple filename patterns to the erase command for the clean
    targets.  This is not allowed.
        wxstring, gague, itsybits, fafa.
```

3.4.6. Why do I get "data segment overflow error" when linking using Borland compiler?

You have a number of switches in compiler configuration that may need adjusting. In order of my own preference:

1. Make sure that stack is located in a different segment from data. It saves you about 16k.

   IDE:

- "Assume DS==SS" - Never, Never and Never!
- "Smart callbacks" -Off.

Command line:

- -WE -Fs-

DrawBacks: All functions that are exported to Windows must be explicitly declared so, and MakeProcInstance() call is neccessary. (This is done in wxWin, just take care of your own callbacks).

For version 150k you have to uncomment one MakeProcInstance() in src/msw/wx_main.c (or wx_win.c - don't remember exactly). The patch is sent to Julian.

2. Put virtual tables into code segment. It saves you 4bytes*per_virtual_function_per_class. Inherited functions also count. Result is more than you can expect.

IDE:

- "Far virtual tables" - On.

Command line:

- -Vf

Drawbacks: All C++ libraries and objects must be compiled with the same settings of this switch or you'll get "undefined symbol" link errors. Put this is in your default configuration file and recompile everything to save yourself from trouble later.

3. Turn on automatic far data.

IDE:

- "Automatic far data" - On,
- "Far data threshold" - 4.

Command line:

- -Ff=4

Drawbacks: you can't run more than one instance of your program anymore. This is architectural flaw in MS-Windows. If you need to, make several copies of your .exe under different names and run one copy of each.

4. Put constant strings in code segments.

IDE:

- "Put constant strings in code segments"

Command line:

- -dc

Drawbacks: Not available with Borland 3.1. You probably wan't be able to modify any of these strings, since they are in code segments. If windows does not prohibit write access to code segments, then this will cause hard to find bugs. (Code segments are marked as discardable and can be thrown away and reloaded from .exe file at any moment). (I did not tried it myself yet).

Contributed by Alexei Vovenko <alv@au.edu.dstc>

3.4.7. My Windows compiler doesn't contain all the required .lib files.

Some compilers seem to be shipped without libraries such as ddeml.lib, shell.lib, and commdlg.lib. Use implib to generate corresponding .lib files, e.g.

```
implib commdlg.lib c:\windows\system\commdlg.dll
```

3.4.8. Is it possible to use multithreaded library with wxWindows under MS-Windows?

Yes, if you do it carefully. First of all make sure that your compiler does not make any unreasonable assumptions about stack segment, and better still it makes no assumptions about stack segment at all. Make sure that both the library and your program are compiled this way.

Do not expect wxWindows to be reentrant because MS-Windows is not. Most probably your thread implementation does not use preemptive scheduler, so MS-Windows reentrancy is not a big problem.

Derive your member from "Bool wxApp::OnIdle(void)" and put a pthread_yield() call there. If you use dialogs, take care about yielding to threads while in a dialog.

Contributed by Alexei Vovenko <alv@au.edu.dstc>

3.4.9. Can I use wxWindows with Watcom C++?

Yes, Watcom C++ can be used in WIN386 (32-bit) mode and WIN32s mode. However, there seems to be a problem running wxWindows programs compiled in 16-bit mode.

To compile with Watcom in WIN386, set USE_ODBC to 0 in wx_setup.h. All the other settings should work. A fix to allow the ODBC classes to be used with Watcom (similar to that which allows CTL3D to be used) will probably be released shortly.

Don't forget to set the INCLUDE and PATH variables appropriate to the mode of compilation. For WIN386 compilation, these are:

```
PATH F:\WATCOM\BIN;F:\WATCOM\BINW;F:\WATCOM\BINB;<the rest of your
path>
SET INCLUDE=F:\WATCOM\H;F:\WATCOM\H\win
```

For WIN32 (NT) target compilation:

```
PATH F:\WATCOM\BINNT;F:\WATCOM\BINB;F:\WATCOM\BINW;F:\WATCOM\BIN;<the
rest of your path>
SET INCLUDE=F:\WATCOM\H;F:\WATCOM\H\NT
```

Edit the file wx/src/makewat.env and set the WATCOM, WXDIR and MODE variables appropriately (see comments).

You will probably want to set up some batch files to switch between settings, if you are targetting more than one platform. If you can't find the BINNT directory, you probably didn't check the Windows NT target option when installing. The naming is confusing, since you're probably not targetting 'genuine NT', but WIN32s.

Under WIN32s (NT) compilation, we unfortunately lose CTL3D and ODBC capabilities. There may be a way of linking with libraries: watch this space. When running the hello.exe demo in WIN386, copying a metafile to the clipboard produces a fairly catastrophic crash (often exiting Windows or hanging completely). The culprit appears to be wxMetaFile::SetClipboard and the calls to GlobalAlloc, GlobalLock and the subsequent memory access. All the other functionality of hello.exe appears to work under Watcom C++, and the crash does *not* occur in WIN32s mode.

Results with some of the other samples in WIN386 and WIN32s modes:

- The animate sample doesn't show the animation in WIN386 mode, but it works in WIN32 mode.
- The toolbar sample works fine in WIN386 and WIN32 modes.
- The buttnbar sample crashes Windows in WIN386 mode (the underlying buttonbar code is 16-bit Windows code taken from a Microsoft sample: probably needs sanitizing). It works fine in WIN32 mode.
- The dialogs sample works fine in WIN386 and WIN32 modes.
- The docview sample crashes Windows when print-previewing in WIN386 mode. It works fine in WIN32 mode.
- The form sample works fine in WIN386 and WIN32 modes.
- The fractal sample works fine in WIN386 and WIN32 modes.
- The ODBC sample can't be compiled yet since we can't link 32-bit apps with 16-bit ODBC.
- The layout sample works fine in WIN386 and WIN32 modes.
- The mdi sample works fine in WIN386 and WIN32 modes.
- The minimal sample works fine in WIN386 mode and WIN32 modes.
- The types sample works fine in WIN386 and WIN32 modes.
- The panel sample crashes windows in WIN386 mode. I suspect the gauge code, again Microsoft-derived 16-bit code. It works fine in WIN32 mode.

3.4.10. How can I compile PROLOGIO with Borland C++?

Borland users have been finding that it's impossible to compile the LEX and YACC-generated parser files in PROLOGIO from wxWindows 1.50. PROLOGIO is required for compiling wxBuilder.

Peter Trattler and Edward Zimmermann have improved PROLOGIO to allow FLEX to be optionally used instead of LEX; see the PROLOGIO manual for more details. If you have FLEX (the default Linux lexical analyser generator), you can generate a lex_yy.c which compiles more cleanly.

LEX and FLEX-generated versions of lex_yy.c are supplied, as lexyy.c and flexyy.c respectively.

FLEX and YACC are freely available for DOS; most major DOS ftp sites should have them.

Remember *not* to compile lex_yy.c explicitly. It's included by y_tab.c.

3.4.11. I get a compilation error in wb_item.cpp using Borland C++.

In release 1.50 k there's a bug: edit wb_item.cpp, go down to the wxbRadioBox constructor and replace

```
#ifndef _turboc
```

with

```
#ifndef __BORLANDC__
```

3.4.12. Can I use DJGPP with wxWindows?

No. Although there is now a Windows 3.1 version of RSXDK which allows limited 32-bit Windows compilation under Windows using DJGPP, it doesn't have enough functionality to be useful for wxWindows. This has been checked out by Arjen Duursma (arjen@capints.uucp).

3.4.13. Can I use GNU-WIN32 with wxWindows?

Yes. Thanks to Keith Garry Boyce, from 1.66E onwards wxWindows supports GNU-WIN32, with a few patches to the compiler headers. See install.txt for more details.

3.4.14. Can I make a DLL out of wxWindows to reduce executable size?

Unfortunately, this is not yet possible. However, an attempt has been made for VC++ 4.0. The following reproduces the file docs/dll.txt.

```
 Attempt to create a DLL of wxWindows using VC++ 4.0
 ---------------------------------------------------

 The wxWindows source has been changed for a nearly (!) successful
 attempt to create a DLL out of wxWindows, in order to substantially
reduce
 executable size (and link time).

 Perhaps someone else will be able to fathom what I'm doing wrong.

 To compile for DLL
 ------------------

 Edit wx_setup.h and src/msw/makefile.unx and set
 MINIMAL_WXWINDOWS_SETUP to 1 in both files.

 Compiling library:

 > cd c:\wx\src\msw
 > nmake -f makefile.nt dll     ; Makes c:\wx\lib\wx166.dll/.lib
```

```
 > nmake -f pch                 ; Makes dummy.obj and wx.pch for apps
                                ; to use

 Compiling samples:

 Edit samples/minimal/makefile.nt and set WXUSINGDLL to 1 just before
 including ntwxwin.mak.

 > nmake -f makefile.nt         ; Make minimal.exe
 > copy c:\wx\lib\wx166.dll .
 > minimal.exe

 Similarly for hello.

 What goes wrong
 ---------------

 When running minimal.exe (which is a gratifying 80K or so in size),
 the button and message do not appear.

 On tracing the code, it seems that in SetSize, a call to
 GetWindowText to get the message or button label, doesn't work.
 It copies zero characters to the buffer. Therefore the size
 is miscalculated and the height is zero - so the items don't show.
 The extended error code is 120, which means 'This function is
 only valid in Win32 mode', according to winerror.h.

 The mystery is why this function should not work. Hunting through
 on-line help, there is a comment about Set/GetWindowText not working
 across applications (but we're not using it across apps) and
 something in the Knowledge Base about SetWindowText not working
properly
 across threads. Nothing very relevant for DLL usage.

 When running hello.exe, there's a kernel GPF when calling the
 constructor for the subwindow. Unfortunately there's absolutely
 no indication of what the error might be caused by.

 Notes on making the DLL
 -----------------------

 I've added a WXDLLEXPORT definition (defined in wx_defs.h) and
 sprinkled class and function declarations liberally with this
 keyword, as per VC++ documentation. This changes according to
 whether the DLL is being made (WXMAKINGDLL defined) or being
 used by an app (WXUSINGDLL defined).

 For an app to indicate that it's using the DLL version instead
 of the statically linked version, you just need to put

   WXUSINGDLL=1

 in the makefile _before_ you include ntwxwin.mak.

 Note that the symbols defined in wb_data.obj remain undefined
 when linking with wx166.lib, so I've had to duplicate them
 in dummy.cpp, whose object will be linked to the application in order
```

```
to conform with precompiled header rules. Perhaps someone can
find a way of eliminating this requirement - possibly the module
is not linked properly because it doesn't have any functions in it?
Similarly, WinMain has to be defined in an object linked statically
to the app, so this is in dummy.cpp too.

If anyone can finish the job, I'll be delighted and will ensure that
future versions of wxWindows will be DLL-friendly, at least for
32-bit VC++.

Good luck!

Julian Smart
2nd October 1996
```

3.4.15. Link errors with VC++ 4.2

You need to edit src/ntwxwin.mak. Remove /NODEFAULTLIB from WINLINKFLAGS, and remove libc.lib from WINLIBS.

4. C++ issues

4.1. How can I have class member functions as callbacks for buttons?

Here's some correspondence on the subject.

```
Date: Thu, 18 Nov 1993 11:15:44 +0200
From: Syst{ Kari <ks@fi.tut.cs>
Message-Id: <199311180915.AA06005@karikukko.cs.tut.fi>
To: wxwin-users@ed.aiai, tane@fi.uta.cs
Subject: Re: member function as wxFunction

> > is it true that i can't use a member function of a class as a
callback
> > function?
>
> Yes, unless you declare your member function as static. But then you
> don't have this-pointer available in your function.

There is actually a way. The I first define a new button class:

void my_text_callback (wxObject &obj, wxEvent &ev);

// The typedef for a callback member.
// I found something like this from the Solbource OI include files,
// I just copied it without fully understanding the corresponding
// C++ rule.
typedef void (wxObject::callback_member_func)(...);

// A class with a possibility to send callback method calls to other
// objects/
class MyButton: public wxButton {
public:
  wxObject *client_obj;
  callback_member_func* mem_func;
  MyButton(wxPanel *panel, wxObject *client,
       callback_member_func *mem, char *label) :
  wxButton(panel, my_text_callback, label)
    {
    client_obj = client;
    mem_func = mem;
  };
};

void my_text_callback (wxObject &obj, wxEvent &ev) {

/* This is a terrible glude. The member 'mem_func' should
   should defined at the top of inheritance hierarchy */
  MyButton &o = (MyButton &) obj;
  wxObject *to = o.client_obj;
  (to->*(o.mem_func))(obj,ev);
};

Now, I can set a call back which is a object+method pair (see "new
MyButton"):
```

```
#include <stream.h>
#include <wx.h>
#include "mybutton.h"

wxText *text;

class MyText: public wxText { // A new class with a call-back method
  int n;
  static char *toolkits[4];
public:
  MyText(wxPanel *p, wxFunction f, char *l) : wxText(p,f,l) {
    n = 0;
    this->SetValue(toolkits[0]);
  };
  void Inc() {
    ++n;
    if (n >= sizeof(toolkits) / sizeof(toolkits[0]))
     n = 0;
    this->SetValue(toolkits[n]);
  }
};
char *MyText::toolkits[4] = {"wxwin", "uit", "oi", "interviews"};

class MyApp: public wxApp {
public:
  wxFrame *OnInit() {
    wxFrame *frame = new wxFrame(NULL,"Button test");
    wxPanel *panel = new wxPanel(frame);
    text = new MyText(panel, NULL,"Toolkit");
    panel->NewLine();

    new MyButton(panel, text, &MyText::Inc, "Push Me2");
    frame->Show(TRUE);
    return frame;
  };
} myApp;

From: Burell David Kingery <kingery@edu.tamu.cs>
Message-Id: <9312171452.AA05907@clavin>
Subject: wxWindows code sample
To: Craig Cockburn <craig@ed.festival>
Date: Fri, 17 Dec 1993 08:59:15 -0600 (CST)

Craig,

Here is the first code sample I promised.  It is a list dialog which
can swap
between two different lists.  In my application I use it to swap
between
active and inactive employees (those still working and those who
aren't).

It is a good example of how to use static member functions as
wxFunctions
for callbacks.

Here is the .h file for the list dialog.  I have added comments where I
```

```
thought they were appropriate so search for //= to find them.

Hope this helps,

David

================================ CODE STARTS HERE
=========================
/* listdiag.h */

#ifndef LISTDIAG_H
#define LISTDIAG_H

class ListDialog
{
  public:
    ListDialog(wxFrame *parent, int x, int y, char *message,
             char *caption, int count1, char *list1[],
             int count2, char *list2[], char **choice, int *index);
    ~ListDialog(void);

  private:
    wxDialogBox *dialog;
    wxListBox     *listbox;
    wxButton      *list_select;
    char    *the_selection;
    int           the_index;
    int     visible_list;
    int     c1;
    char    **l1;
    int     c2;
    char    **l2;

//=
// Here are the three member functions which actually do the work I
want
// when the callback is invoked.  Since these are member functions they
// will have access to the instance data of the object.

    void    OkCallback(void);
    void    CancelCallback(void);
    void    ListSelectCallback(void);


//=
// Here are three static member functions which are the initial
callbacks
// of the class.  These are invoked in the normal manner for wxWindows.

    static void   OnOk(wxObject& the_object, wxEvent& the_event);
    static void   OnCancel(wxObject& the_object, wxEvent& the_event);
    static void OnListSelect(wxObject& the_object, wxEvent& the_event);
};

#endif
```

```
Now here is the .cc file for the list dialog.  I'm not real sure if you
could
compile and link, but you could give it a try.  I'm now swamped with
other
work here at Texas A&M so it's been awhile since I looked at this code.
Once again search for //= to find pertinent comments.

/* listdiag.cc */

/*
 * standard header goes here.
 */

#include <windows.h>

extern "C"
{
#include <stdio.h>
}

#include "wx.h"

#include "listdiag.h"

ListDialog::ListDialog(wxFrame *parent, int x, int y, char *message,
                   char *caption, int count1, char *list1[],
                   int count2, char *list2[], char **choice, int
*index)
{
  if (x < 0) x = 300;
  if (y < 0) y = 300;

  wxDialogBox the_dialog(parent, caption, TRUE, x, y, 300, 250);

  dialog = &the_dialog;
  dialog->SetClientSize(300, 250);
  (void)new wxMessage(dialog, message);
  dialog->NewLine();

  listbox = new wxListBox(dialog, NULL, NULL, wxSINGLE,
                    -1, -1, 150, 200,
                    count1, list1);
  visible_list = 1;
  c1 = count1;
  l1 = list1;
  c2 = count2;
  l2 = list2;

  dialog->NewLine();

//=
// Here I am setting the callback function of the button to be the
// static member function ListDialog::OnOk.  The other buttons are
// the same way.

  wxButton *ok = new wxButton(dialog,
                        (wxFunction)ListDialog::OnOk, "OK");
```

```
//=
// Here is where I set the client data of the button to the this
pointer.
// This will be used in the static member function to invoke the
correct
// method on the correct object.

  ok->SetClientData((char *)this);


  wxButton *button = new wxButton(dialog,
                          (wxFunction)ListDialog::OnCancel,
                          "Cancel");
  button->SetClientData((char *)this);
  list_select = new wxButton(dialog,
                       (wxFunction)ListDialog::OnListSelect,
                       "Show Inactive");
  list_select->SetClientData((char *)this);

  ok->SetDefault();

  dialog->Fit();
  dialog->Centre();
  dialog->Show(TRUE);
  *choice = the_selection;
  *index = the_index;
}

ListDialog::~ListDialog(void)
{
  return;
}

void ListDialog::OnOk(wxObject& the_object, wxEvent& the_event)
{
  ListDialog      *list_dialog;

//=
// Here is a static member function which is used as a wxFunction.
// The wxObject is the button which was pressed and I use it to get
// the client data of the button.  Remember the client data is the
// this pointer of the object of interest.

  list_dialog = (ListDialog *)((wxButton&)the_object).GetClientData();

//=
// Now that I have the correct object, I can invoke the member function
// of the object.

  list_dialog->OkCallback();

  return;
}

void ListDialog::OnCancel(wxObject& the_object, wxEvent& the_event)
{
```

```
  ListDialog      *list_dialog;

  list_dialog = (ListDialog *)((wxButton&)the_object).GetClientData();

  list_dialog->CancelCallback();

  return;
}

void ListDialog::OnListSelect(wxObject& the_object, wxEvent& the_event)
{
  ListDialog      *list_dialog;

  list_dialog = (ListDialog *)((wxButton&)the_object).GetClientData();

  list_dialog->ListSelectCallback();

  return;
}

void ListDialog::OkCallback(void)
{
//=
// This is the member function of the object which gets invoked from
the
// static member function ListDialog::OnOk.  Now that I'm within a
member
// function I can access the instance data of the object as well as
invoke
// other member functions.  The other callbacks are handled the same
way.

  the_index = listbox->GetSelection();

  if (the_index != -1)
    {
     the_selection = copystring(listbox->String(listbox-
>GetSelection()));
    }
  else
    {
     the_selection = NULL;
    }

  dialog->Show(FALSE);
  return;
}

void ListDialog::CancelCallback(void)
{
  the_selection = NULL;
  the_index = -1;
  dialog->Show(FALSE);
  return;
}

void ListDialog::ListSelectCallback(void)
```

```
{
  listbox->Clear();

  switch (visible_list)
    {
     case 1:
       listbox->Set(c2, l2);
       list_select->SetLabel(" Show Active ");
       visible_list = 2;
       break;

     case 2:
       listbox->Set(c1, l1);
       list_select->SetLabel("Show Inactive");
       visible_list = 1;
       break;
    }

  return;
}
```

```
Date: Wed, 12 Mar 1997 16:29:36 -0500
From: Bill McGrory <mcgrory@aerosft.com>
To: Julian Smart <julian.smart@ukonline.co.uk>

Hi Julian,

The examples you have of using member functions in callbacks
needs updating for what I believe is ANSI C++.
(the file members.txt, and the html page on c++ issues)

I tried the first example, and that failed, but with a little playing
around I got something to work.

here's my sample code. (it compiles with SGI's c++ compiler, and
Digital Unix's as well)

#include "stdio.h"

class CallbackClass;

typedef void (CallbackClass::*Func)();

class CallbackClass
{
private:
      int   junk;
};

class Main: public CallbackClass
{
public:
      int         data1, data2;

      void  Callback1(void) {printf("data1 = %d\n",data1);};
      void  Callback2(void) {printf("data2 = %d\n",data2);};
```

```
};

class Second
{
public:
      CallbackClass     *theObj;
      Func  theFunc;

      void  EvalCallback(void);
};

void
Second::EvalCallback(void)
{
      (theObj->*theFunc)();
}

main()
{
      Main  *main1, *main2;
      Second      *second;

      main1 = new Main;
      main2 = new Main;

      main1->data1 = 1;
      main1->data2 = 2;
      main2->data1 = 3;
      main2->data2 = 4;

      second = new Second;

      second->theObj = main1;
      second->theFunc = (Func)&Main::Callback1;
      second->EvalCallback();

      second->theObj = main1;
      second->theFunc = (Func)&Main::Callback2;
      second->EvalCallback();

      second->theObj = main2;
      second->theFunc = (Func)&Main::Callback1;
      second->EvalCallback();

      second->theObj = main2;
      second->theFunc = (Func)&Main::Callback2;
      second->EvalCallback();
};
```

4.2. How can I display debugging messages?

If you use the function wxDebugMsg, messages will be displayed on either the standard error stream (X) or the debugging stream (Windows). To read these messages under Windows, you must either be running a debugger, or (perhaps more convenient), using a program such as Microsoft's DBWIN which displays these debugging messages in a text window.

Using DBWIN on a regular basis has the advantage of showing up some GDI errors that you might otherwise miss, but that could cause severe problems in future. DBWIN is available in the Windows SDK or on the Developer's Network CD-ROM, and probably by ftp from Microsoft's site.

DBWIN doesn't work under Windows 95 or Windows NT, unfortunately.

wxWindows supports debug logs: see the documentation for wxDebugContext.

5. Platforms

5.1. Is there a Mac version of wxWindows under development?

There is a volunteer port in progress and alphas can be downloaded from the AIAI ftp site. An official AIAI Mac port is due to be completed in early July 1996.

5.2. Is there an X version of wxBuilder?

Sort of... the XView version needs some work, particularly because there are occasions when 2 levels of modal dialogs are used, which XView doesn't like.

The Motif version is coming on well and will be released (a binary for Suns and HPs) by the end of September.

6. Run-time problems

6.1. How do I install CTL3DV2.DLL correctly?

A program that uses CTL3DV2.DLL must be installed so that the DLL is in the windows/system directory, and NOT in the application directory, or it will not run correctly.

It is tempting to copy CTL3DV2.DLL into as many directories as possible to hedge your bets. Unfortunately this is exactly the wrong thing to do: the DLL must be in windows/system, ONLY.

CTL3D.DLL is not required if you're linking with CTL3DV2.LIB: it's the old version.

6.2. Why does my program exit abnormally when initializing?

You may be declaring a pen, brush, icon, cursor or colour globally. These objects automatically add themselves to global lists which may not be initialized before the object constructors are called, and so only global pointers to these objects may be declared. After or during **OnInit** is called, these objects may be created with impunity.

6.3. After using memory DCs and bitmaps under Windows, I get system crashes.

Here's some correspondence that helps explain this phenomenon and its solution.

```
From: Markus Meisinger <Markus.Meisinger@at.ac.uni-linz.risc>
Message-Id: <199406241203.AA09143@melmac.risc.uni-linz.ac.at>
To: wxwin-users@ed.aiai
Subject: SOLUTION to strange problems of yesterday
Status: RO

hi wxWindows programers

i wrote yesterday:

>I have encountered some strange problems with the sample programs
under
>MS-WINDOWS 3.11. (they work fine under UNIX/LINUX)
>
>The problems arise if i e.g quit the minimal sample program. During
the run
>of the program no problems appear. But after it stops than the whole
>system crashes (not always, but quite often :)!
>
>If the system crashes, not only the sample program produces an
>UNRECOVERABLE APPLICATION ERROR, but also the most other programs
running
>at the same time (PROGMAN, FILEMAN, WINMETER, ...) stop their work
with
>a severe error. :(
>
>Any help would be greatly appreciated! Or is there a patch available,
because
>a guess their is a bug in freeing the system resources ...

The actual problem causing the strange behaviour was a buggy wxWindows
```

```
program
of mine running some minutes before and not the sample programs, CTL3D
or
FAFALIB

Because i used a wxMemoryDC in which i selected a dynamicaly created
bitmap.
During the quiting process i freed this bitmap which was selected in
the
wxMemoryDC. But wxWindows also freed this bitmap during deleting the DC
and
... the rest you know

SO, BE AWARE IF YOU DELETE BITMAPS!

Never delete a bitmap which is selected into a DC if the DC will be
deleted.
The bitmap will be deleted by the DC!
But i did it !!! :) and it resulted in a system wide crash of MS-
WINDOWS
(in Motif it works fine (why?))
The funny thing with this crash is that the system doesn't immediately
crash,
instead it works fine for a few minutes until ANY other program gets
into
some memory conflict. After this application error each program still
running
complains problems with the memory and stops. The last thing you see is
the standard windows background color (if you are lucky:)
I never saw such a system wide crash WOU :) in some sense it was funny
:)

Hope you don't make the same errors as i did!

Markus
```

6.4. Why does my canvas not have the keyboard focus under Windows?

Under MS Windows, if you have more than one subwindow per frame, you need to call wxWindow::SetFocus explicitly, or wxWindows will not know which subwindow needs the focus. A good place to do this is in wxFrame::OnActivate.

6.5. Why do panel items not size or position correctly under Motif?

Absolute positioning in Motif doesn't mix well with constraint-based positioning, as used by wxWindows to implement left-to-right, top-to-bottom layout.

If you know that all your widgets are going to be positioned and sized explicitly, you should pass the style flag **wxABSOLUTE_POSITIONING** to the panel constructor, in which case a Motif bulletin board widget will be used instead of a form widget.

NOTE: from wxWindows 1.60, the wxABSOLUTE_POSITIONING flag is obsolete, because all Motif positioning is now done this way.

One reason for a widget not appearing can be that the widget is sized just too big for the panel, and then Motif gets very confused. Try reducing the size of the widget.

Another possibility is to try setting the widget sizes first, and finally setting the panel size.

Widgets that are too close together may cause problems.

6.5.1. Possible cure for some Motifs

Some Motifs under some platforms appear to have very bad panel item problems in dialogs. Once cure seems to be to change the resize policy in src/x/wx_dialg.cpp: find where two lines are marked TROUBLE SPOT (from 1.61 beta 2), in the 'else' clause of the 'invisibleResize' condition. One line has XmRESIZE_ANY, the second has XmRESIZE_NONE.

Comment out or uncomment the first line, and uncomment or comment out the second line (whichever is not already the case). However, doing this seems to make a dialog box's size change dynamically when (for example) a listbox next to the dialog box edge is updated; so only change this if absolutely necessary.

6.6. Under Motif, the status line does not appear.

The fix seems to be to call Fit before calling CreateStatusLine, or try calling SetSize after frame creation.

6.7. Under Windows, dialog boxes refuse to appear. Why?

You probably forgot to include the file wx.rc in your resource script This is required for dialog boxes to work correctly.

6.8. Under Motif, quitting windows from the File menu causes a crash.

There is a bug in the X toolkit that manifests itself on some platforms.

Here's a bug fix from Jim Huggins:

```
I've been investigating this problem for the last few
weeks.  (where the problem is:  closing child window
in motif causes core dump.)

The core dump stack is usually:

>0  0x3ff80c5f8e4 in InSharedMenupaneHierarchy() RowColumn.c:6759
#1  0x3ff80c5fc44 in SetCascadeField() RowColumn.c:6849
#2  0x3ff80c6d924 in MenuProcedureEntry() RowColumn.c:11685
#3  0x3ff80bd0a88 in Destroy() CascadeBG.c:2350
#4  0x3ff803c9d3c in Phase2Destroy() Destroy.c:113
#5  0x3ff803c9c04 in Recursive() Destroy.c:72
#6  0x3ff803c9b94 in Recursive() Destroy.c:60
#7  0x3ff803ca0b8 in XtPhase2Destroy() Destroy.c:205
#8  0x3ff803ca220 in _XtDoPhase2Destroy() Destroy.c:252
#9  0x3ff803d1a90 in XtDispatchEvent() Event.c:1127
#10 0x120026b7c in ((wxApp*)0x1400011b8)->MainLoop() wx_main.cpp:177

>From what I can tell, this is a bug in an older version
of Xt library version X11R5.  I think there was a patch
for it but I have no idea what number or when it was produced.
```

```
The bug has to do with recursive destruction of Menu widgets.
(I don't know too much detail...)

I tried posting a note to comp.windows.x.intrinsics and didn't
get any working fixes or explanations.  (My idea about a Xt bug
and possible patch comes from a note I found in an internal DEC
notes conference.)

I found a work around for this problem in wxWindows though:
in file srx/x/wx_item.cpp,

add the follow code to the bottom of function wxMenu::~wxMenu(void)
(I think around line 1070)

#ifdef wx_motif
  if (handle) {
      DestroyChildren();
      wxWidgetHashTable->Delete((long)handle);
      Widget w = (Widget)handle;
//     XtDestroyWidget(w);
      handle = NULL;
  }
#endif

This is basically doing what the ~wxWin function would have done
except the XtDestroyWidget(w) has been commented out to avoid
the Xt bug.  (i.e. if you uncomment that line above, the core dump
will occur.)

Jim
```

6.9. Under XView, I get a SERVER_IMAGE_BITMAP_FILE warning mesage.

```
XView warning: SERVER_IMAGE_BITMAP_FILE: Server image creation failed
(Server Image package)
```

The application may be looking for an icon file which does not exist or has been referenced by a relative icon pathname. Use an absolute path, or include the icon image in the source file using conditional compilation (see the reference manual's **wxIcon** class entry).

6.10. Under Windows, MDI child windows don't size properly.

For some reason, it's not possible to programmatically resize an MDI child frame just after creation, so using *Fit* to size an MDI child frame around a subwindow does not work.

Also, it does not seem to be possible (on creation at least) to hide an MDI child frame, resize its subwindows, then show it.

If you are using *Fit* to size the child window, you must put up with seeing the windows repaint themselves: the child window will come up already visible.

However, if you know the size of the window in advance, give it an absolute size, create it with the wxMINIMIZE window style, draw the contents, then call *Iconize(FALSE)* to restore the window. This at least avoids seeing half-painted windows whilst things get initialized.

6.11. Functions that return string values cause strange behaviour on some platforms.

To resolve the question of who is responsible for allocating and deallocating memory, wxWindows maintains a policy (unless documented otherwise) of returning a *temporary* pointer to a static string buffer from functions such as *wxText::GetValue*.

If you don't immediately take a copy of this value, it's possible that subsequent wxWindows calls will use the same memory, causing unpredictable results. This tends to be more the case under Windows than other platforms, where pointers to internal XView or Motif widget strings are often returned.

7. What you can't do in wxWindows

There are many exceptions and provisos in the wxWindows API, mostly because the individual platforms don't support some functionality, or it hasn't yet been implemented, or some other reason. These exceptions are being eliminated where possible, but inevitably some will remain.

This section will start to give an at-a-glance guide to what *not* to try, and perhaps save some grief. It is not a complete list though.

Fonts in panel items. XView doesn't allow you to set panel item fonts individually.

wxTextWindow::OnChar. XView doesn't allow interception of character input in a text subwindow.

OnSetFocus, OnKillFocus. Not called for panel items (yet). May not be called for other windows either; beware.

Bitmaps and arcs in PostScript device context. The PostScript device context does not support drawing bitmaps, or arcs.

Menu items cannot be deleted dynamically. Dynamic menu item deletion has not been implemented. It is not technically infeasible on any known platform, though.

Custom cursor creation. Not yet implemented.

Bitmap buttons. Bitmaps loaded dynamically from .BMP or .GIF files under UNIX, cannot be used for bitmap buttons yet. XBM and XPM files should work fine meanwhile.

OLE-2. Not supported; an OLE++ project has been started but is currently in limbo.

Index