wxWidgets/wxX11 version 2.4.2 for nano-X application note

1. wxWidgets/wxX11 overview

wxWidgets is a C++ library that lets developers create applications for Windows, macOS, Linux and other platforms with a single code base. wxWidgets defines a common API across platforms, but uses the native graphical user interface (GUI) on each platform. This way an application using wxWidgets will take on the native look and feel that users are familiar with.

wxX11 is a port of wxWidgets using X11 (The X Window System) as the underlying graphics backend. wxX11 draws its widgets using the wxUniversal widget set. So it does not use the native GUI of the particular platform but implements its own GUI based on X11. It shall work with wxWidgets for Windows, GTK and X11. The X11 port of wxWidgets has not been updated and maintained for a long time. The wxUniversal readme file mentions: „Please note that wxUniversal is not as mature as the other wxWidgets ports and is currently officially in alpha stage. In particular, it is not really intended for the end users but rather for developers“ (in 2002).

Getting wxX11 to work with nano-X requires a number of settings and modifications. The 2.4.2 version has been selected because it is close to the version when the nano-X branch was maintained last.

The X11 FAQ mentions that the following classes and features are not yet implemented: wxSpinCtrl, wxSocket, wxToggleButton, clipboard classes, drag and drop classes, tooltips. The readme-nanox file mentions that the colour database needs to be implemented and the mask capability has to be added, without which controls won't display properly.

Also, only a part of the samples included in the wxWidgets package will run with wxX11.

2. wxX11 running based on nano-X

a) Download
b) Generate required directories
c) Run configure
d) Modify the setup.h file
e) Code changes required
f) Using the NtoNX.h header file
g) Compile the library
h) Compile the minimal example program
i) Run the minimal example program

This are the steps you need to follow, to get wxX11 to run with nano-X installed on your Linux distribution.

a) Download and unpack wxX11-2.4.2 into the wxX11-2.4.2 directory. You can download this version here: https://sourceforge.net/projects/wxwindows/files/

b) Generate a buildnanox directory in the wxX11-2.4.2 directory. Also generate a wxX11-2.4.2/samplesx11 directory to compile the examples from the command line into that.

Also you need to put the XtoNX.h file from the microwindows-0.89pre8 version into the src/include directory of you current microwindows directory, in my case that is microwindows-dev/include. You can download this microwindows version from this link: ftp://microwindows.censoft.com/pub/microwindows/Historic/

c) Get into the buildnanox directory and run configure (in a script) with these parameters:

```
export MICROWIN=/home/georg/microwindows-dev

../configure --with-x11 --disable-shared  --without-odbc --disable-debug \
--disable-xpm --enable-nanox –disable-sockets \
--x-includes=/usr/include/X11 --x-libraries=/usr/lib64 --enable-permissive \
--without-libpng --disable-calendar --disable-dataobj  --disable-mdi \
--disable-snglinst --disable-fontmap --disable-filesystem --disable-tabdialog \
--disable-log --without-sockets --disable-url --disable-html --disable-filesystem \
--disable-fs_inet --disable-fs_zip --disable-printarch --disable-protocols \
--disable-commondlg --disable-propsheet
```

The two dots in front of configure execute the configure script in the wxX11-2.4.2 directory.

The parameters for configure are selected as required for using it with nano-X.

If you run ../configure for the first time it will generate the following directories:
demos, lib, samples, tests and utils plus a Makefile and the wx-config script.

The lib directory contains a setup.h file in the /home/georg/wxX11-2.4.2/buildnanox/lib/wx/include/x11univ-2.4/wx directory. This file is set up by the configure script and contains lots of macros which define the operation of wxWidgets. It can be modified manually as we will have to do.

d) Modify the file setup.h in the buildnanox/lib/wx/include/x11univ-2.4/wx directory.

First generate a symbolic link to the setup.h file in the wxX11-2.4.2/include/wx directory. This link has to point to the buildnanox/lib/wx/include/x11univ-2.4/wx/setup.h file.

Then modify the following macros in the setup.h file:

```
#define wxUSE_FILEDLG 0
#define wxUSE_FONTDLG 0
#define wxUSE_TEXTDLG 0
#define wxUSE_CHOICEDLG 0
#define wxUSE_COLOURDLG 0
#define wxUSE_DIRDLG 0
#define wxUSE_CHOICE 0
```

```
#define wxUSE_SOCKETS 0
#define wxUSE_DOC_VIEW_ARCHITECTURE 0
#define HAVE_FCNTL 1
#define wxUSE_LOG 1
#define wxUSE_LOGWINDOW 1
#define wxUSE_LOGGUI 1
```

e) Code changes required for nano-X

The following code changes are required:

1) in the file src/unix/utilsx11.cpp:
```
#if 0 //defined(__WXX11__) || defined(__WXGTK__) || defined(__WXMOTIF__)
```

2) in the file src/x11/toplevel.cpp line 404:
```
//wxSetIconsX11( wxGlobalDisplay(), GetMainWindow(), icons );
```

3) in the file src/x11/window.cpp:
```
XSetInputFocus( wxGlobalDisplay(), xwindow, 0, CurrentTime );
```

f) Using the NtoNX.h header file:

From the wxX11 readme-nanox file:

Nano-X has a different API from Xlib, although there are many similarities. Instead of changing the wxWindows code to reflect Nano-X conventions, a compatibility layer has been added, in the form of these files:

include/wx/x11/nanox/X11/Xlib.h	*; Xlib compatibility*
include/wx/x11/privx.h	*; Useful macros*
src/x11/nanox.c	*; Xlib compatibility*

There is also an XtoNX.h compatibility header file in Microwindows, which we augment with our Xlib.h and nanox.c.

This XtoNX.h file has to be the one in the Microwindows version 0.89pre8, because the wxX11 code is based on that.

The wxX11-2.4.2/include/wx/x11/private.h file is included in all programs in the src/x11 directory and the src/unix/fontutils.cpp plus the src/generic/dcpsg.cpp files. This header file includes the include/wx/x11//privx.h file which again includes the include/wx/x11/nanox/Xlib.h file. This Xlib.h header file includes NtoNX.h which includes nano-X.h.
The src/x11/nanox.c file implements functions which meanwhile are in implemented in the NX11 library.

g) Compile the library

First set the MICROWIN environment variable. E.g. enter this on the command line:
```
export MICROWIN=/home/georg/microwindows-dev
```

For this enter e.g.: make -j8. It is a big library and you should set the -j parameter to the number of your processors to speed up the compilation. You can determine the number of processors of your computer with the nproc command. Enter nproc on the command line and it will output the number of processors.

Since wxX11 is old, it writes the generated libraries and files into the buildx11/lib directory during compilation. The wx-config script requires these files in the buildx11/lib64 directory. Therefore make a symbolic link in the /home/georg/wxX11-2.4.2/buildnanox directory called lib64 which points to the /home/georg/wxX11-2.4.2/buildnanox/lib directory.

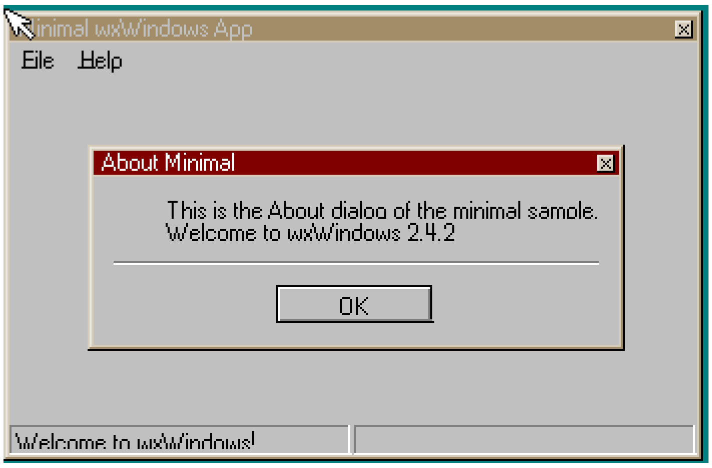
h) Compile the minimal example program

You can compile the example program by entering these parameters using a script like this:

```
curdir=/home/georg/wxX11-2.4.2
libname=x11univ-2.4
g++ ../samples/minimal/minimal.cpp -o ../samplesx11/minimal -g -Wno-write-strings \
-L$curdir/buildnanox/lib64 -pthread \
$curdir/buildnanox/lib64/libwx_$libname.a \
-lnano-X  \
-lpng -ljpeg -ltiff -lz -ldl -lm\
 \
-I$curdir/buildnanox/lib64/wx/include/$libname \
-I$curdir/include -D_FILE_OFFSET_BITS=64 -D__WXUNIVERSAL__ -D__WXX11__
```

i) Run the minimal example program

The minimal program should now be located in the wxX11-2.4.2/samplesx11 directory. To execute it, enter nano-X & ./minimal on the command line in the samplesx11 directory.



This is the About dialog of the minimal sample.
Welcome to wxWindows 2.4.2

Welcome to wxWindows!

3. wxX11 running based on X11

a) Generate required directory
b) Run configure
c) Generate a symbolic link
d) Compile the library
e) Compile the minimal example program
f) Run the minimal example program

This are the steps you need to follow, to get wxX11 to run with X11 installed on your Linux distribution. This way you can compare the execution of the samples using X11 with the nano-X or NX11 versions.

a) Generate a buildx11 directory in the wxX11-2.4.2 directory.

b) Get into the buildx11 directory and run configure with these parameters:

```
../configure --with-x11 --disable-shared  --without-odbc --disable-debug  \
--x-includes=/usr/include/X11 --x-libraries=/usr/lib64 --enable-permissive
```

The two dots in front of configure execute the configure script in the wxX11-2.4.2 directory.

If you run ./configure for the first time it will generate the following
directories:
demos, lib, samples, tests and utils plus a Makefile and the wx-config script.

The demos directory contains a Makefile and the
bombs, forty, fractal, life and poem directories. These contain Makefiles which
will compile the equivalent demos in the "/home/georg/wxX11-2.4.2/demos" directory.
So if you execute the Makefile it will execute the Makefiles in these directories
and the demos will be compiled for wxX11. Currently only the life and poem demos will compile successfully. A demos folder will be generated in the buildx11 directory and the executables are placed there.

The lib directory contains a setup.h file in the
/home/georg/wxX11-2.4.2/buildx11/lib/wx/include/x11univ-2.4/wx directory. This file is set up by the configure script and contains lots of macros which define the operation of wxX11. It can be modified manually if required.

The samples directory contains Makefiles which point to the "/home/georg/wxX11-2.4.2/samples" directories. So, after compiling the library as described below, if you execute the Makefile by entering "make -k", it will execute the Makefiles in these directories and the samples will be compiled for wxX11. Same with the test and utils directories. Use make -k“ since some samples may not compile and the Makefile will terminate prematurely. A samples folder will be generated in the buildx11 directory and the executables are placed there.

c) Generate a symbolic link in the wxX11-2.4.2/include/wx directory. This link has to point, to the /home/georg/wxX11-2.4.2/buildx11/lib/wx/include/x11univ-2.4/wx/setup.h file.

d) Compile the library

Since wxX11 is old, it writes the generated libraries and files into the buildx11/lib directory during compilation. The wx-config script requires these files in the buildx11/lib64 directory. Therefore you need make a symbolic link called lib64 which points to the /home/georg/wxX11-2.4.2/buildx11/lib directory.

To compile enter e.g.: make -j8. It is a big library and you should set the -j parameter to the number of your processors to speed up the compilation. You can determine the number of processors of your computer with the nproc command. Enter nproc on the command line and it will output the number of processors.

On my Linux distribution, there is already a wxWidgets library installed in /usr. Therefore I did not run „sudo make install“.

e) Compile the minimal example program

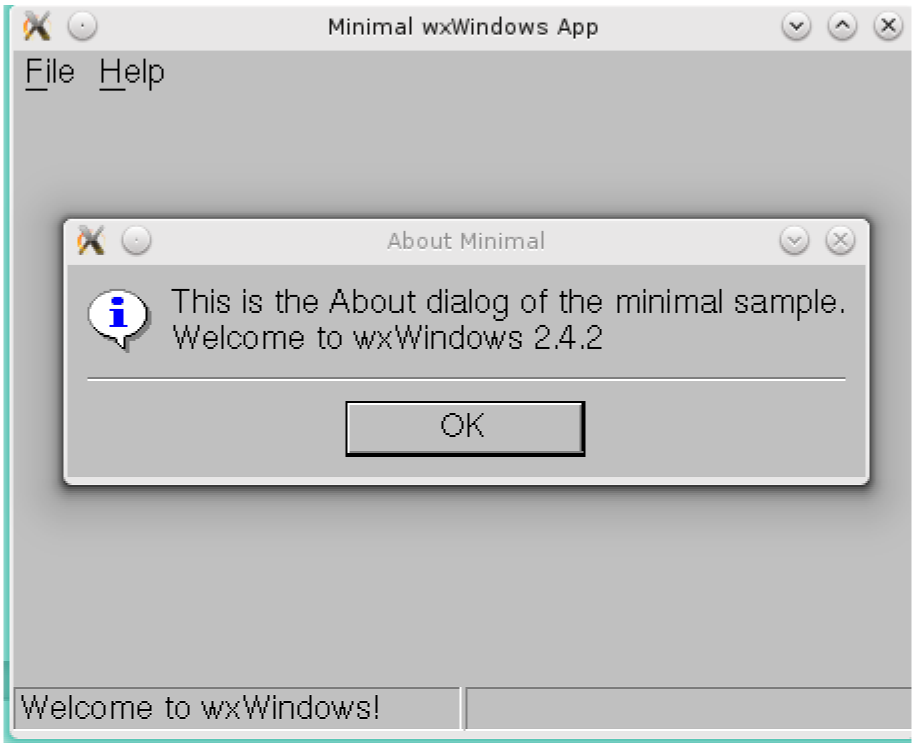
It you enter wx-config --libs core and wx-config --cxxflags on the command line, you should see the required parameters.

You can compile the example program without wx-config by entering these parameters on the command line like this:

```
curdir=/home/georg/wxX11-2.4.2
libname=x11univ-2.4
g++ ../samples/minimal/minimal.cpp -o ../samplesx11/minimal -g -Wno-write-strings \
-L$curdir/buildx11/lib64 -pthread \
$curdir/buildx11/lib64/libwx_$libname.a \
-lX11 -lXpm \
-lpng -ljpeg -ltiff -lz -ldl -lm\
 \
-I$curdir/buildx11/lib64/wx/include/$libname \
-I$curdir/include -D_FILE_OFFSET_BITS=64 -D__WXUNIVERSAL__ -D__WXX11__
```

f) Run the minimal example program

The compiled minimal program should now be located in the wxX11-2.4.2/samplesx11 directory. Click on the file in your file explorer or enter ./minimal on the command line in the samplesx11 directory.



This is the About dialog of the minimal sample.
Welcome to wxWindows 2.4.2

Welcome to wxWindows!

4. wxX11 running based on NX11

a) Generate required directory
b) Run configure
c) Generate a symbolic link
d) Modify the setup.h file
e) Run the replace-lX11.sh script
f) Code changes required
g) Compile the library
h) Compile the minimal example program
i) Run the minimal example program

This are the steps you need to follow, to get wxX11 to run with NX11 installed on your Linux distribution:

a) Generate a buildnx11 directory in the wxX11-2.4.2 directory.

b) Get into the buildnx11 directory and run configure with these parameters:

```
../configure --with-x11 --disable-shared  --without-odbc --disable-debug  \
--x-includes=/usr/include/X11 --x-libraries=/usr/lib64 --enable-permissive \
--disable-sockets --disable-xpm --disable-fs_inet
```

The two dots in front of configure execute the configure script in the wxX11-2.4.2 directory.

c) Generate a symbolic link to the setup.h file in the wxX11-2.4.2/include/wx directory. This link has to point to the buildnx11/lib/wx/include/x11univ-2.4/wx/setup.h file.

d) Copy the replace-lX11.sh script from the microwindows/src directory into the wxX11-2.4.2/buildnx11 directory. Then run „replace-lX11.sh src/make.env“ in the wxX11-2.4.2/buildnx11 directory.

e) Code changes required for NX11

The current version of Microwindows on GitHub includes some Xinerama support.

If you have an older version you have to remove the calls to that library in the wxWidgets code. This is done in the src/unix/displayx11.cpp file at lines 124 and 283, so modify these as below:

```
m_screens = NULL; //XineramaQueryScreens((Display *)wxGetDisplay(), &m_num);
```

and

```
   /* if ( XineramaIsActive((Display*)wxGetDisplay()) )
      return new wxDisplayFactoryX11; */
```

These code changes are not required for standard X11 but we target NX11 as the platform.

f) Compile the library

Since wxX11 is old, it writes the generated libraries and files into the buildnx11/lib directory during compilation. The wx-config script requires these files in the buildx11/lib64 directory. Therefore you need make a symbolic link called lib64 which points to the /home/georg/wxX11-2.4.2/buildnx11/lib directory.

To compile enter e.g.: make -j8. It is a big library and you should set the -j parameter to the number of your processors to speed up the compilation. You can determine the number of processors of your computer with the nproc command. Enter nproc on the command line and it will output the number of processors.

h) Compile the minimal example program

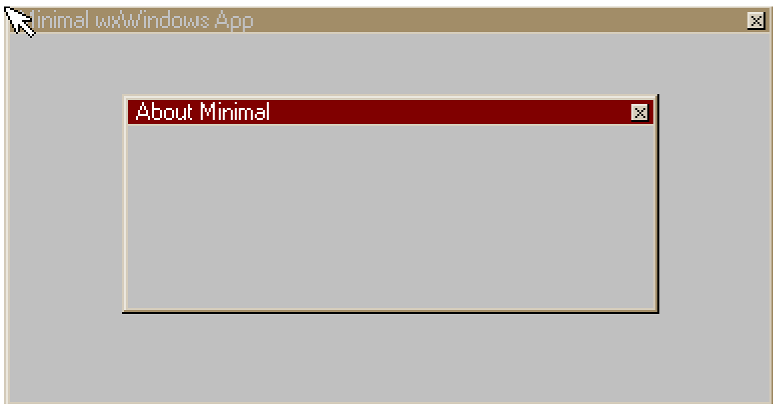
To test if the setup and the libary works, try to compile the minimal example program in the wxX11-2.4.2/samples folder from the command line:

```
curdir=/home/georg/wxX11-2.4.2
libname=x11univ-2.4
g++ ../samples/minimal/minimal.cpp -o ../samplesx11/minimal -g -Wno-write-strings \
-L$curdir/buildnx11/lib64 -pthread \
$curdir/buildnx11/lib64/libwx_$libname.a \
-lNX11 -lnano-X \
-lpng -ljpeg -ltiff -lz -ldl -lm\
 \
-I$curdir/buildnx11/lib64/wx/include/$libname \
-I$curdir/include -D_FILE_OFFSET_BITS=64 -D__WXUNIVERSAL__ -D__WXX11__
```

i) Run the minimal example program

The minimal program should now be located in the wxX11-2.4.2/samplesx11 directory. To execute it, enter nano-X & ./minimal on the command line in the samplesx11 directory.

This program works but does not display text or lines. You also cannot terminate it by clicking on the close button in the upper title bar.



27th August 2019 Georg Potthast