Perl-Gtk3 and Windows

How to install perl-Gtk3 on Windows and create an installer for your perl-Gtk3 programms

1) Installation of the MSys2 Environment

To create a perl-Gtk3 Application and an installer under Windows, you have to install the MSys2 environment first.

Therefor download the x86_64 setup file from www.msys2.org and follow the instructions on the website. Especially don‘t forget to update the package database and the core system (for further information see www.msys2.org).

2) Install the perl-Gtk3 module

Next you have to install the perl-Gtk3 module. This admittedly is a little bit laborius, because we will install the perl modules later again in our alternative root directory. But this makes things easier. And by the way you get a wonderful test environment for your perl-Gtk3 applications!

a)

First install perl with

```
pacman -S --needed --noconfirm mingw-w64-x86_64-perl
```

b)

To install the further needed libraries and modules, we can easily use the perl-Gtk3 installation script:

Therefor download the perl script from https://github.com/MaxPerl/perl-Gtk3-windows-installer and install the perl-Gtk3 module with

```
./install-perl-Gtk3-mingw64.pl
```

3) Create the installer

After everything was installed, you can create the windows installer for your application. We will demonstrate this here with a little perl-Gtk example script from my perl-Gtk3 tutorial (https://github.com/MaxPerl/perl-Gtk3-Tutorial). So our „application“ consists here only of one file (hereinafter called MyApp.pl) which we can easily „install“ by copying it to the right place and making it executable with *pl2bat* (see below).

But also more complex applications shouldn‘t be a problem. If your application for example is distributed with ExtUtils::MakeMaker, you can install it by adding the follwing command in the install-perl-Gtk3-mingw.pl script after the line „$exitcode = install_perl_module(‚Gtk3‘);“ (not implemented yet!!! Only modules from CPAN

can be installed on this way at the moment!):[1]

install_perl_module('MyApp::Module');

Nevertheless here we use a single file-application which we later install by copying it to the right place and make it executable with *pl2bat* (see below).

a)

The installer is created by a bash-script with the help of the QT Installer Framework. You can download all files that we will use here at https://github.com/MaxPerl/perl-Gtk3-Tutorial/tree/master/create_windows_installers_template. The files originate from the gedit windows installer (see https://blogs.gnome.org/nacho/2014/08/01/how-to-build-your-gtk-application-on-windows/ and https://git.gnome.org/browse/gedit/tree/win32). Another inspiration was the MSys2 installer (see https://github.com/msys2/msys2-installer). I only tweak these files on single places.[2]

The gedit guys in the meantime changed to WiX. Nevertheless we will keep using the QT installer framework here, because I like it more and MSys2 contain it out-of-the-box[3]

To create the installer, you only need to make few adjustments:

(1)

In the *make-installer* bash script adjust the variables at the beginning:

```
_version        # the version of your application

_filename       # the name of your setup file which we will create

_builddir       # the „fake-root“ directory, in which our application will be built, before
                  the installer is packed

_perlgtkinstallscript   # the location of the install-perl-Gtk3-mingw64.pl script
```

(2)

After that adjust the function *install_application_packages()* in the *make-installer* bash script, so that all necessary libraries, modules and files will be installed in the fake-root directory. For our example application only few commands are needed:

- First we install perl with:

```
pacman -S mingw-w64-x86_64-perl --noconfirm --root "${_builddir}"
```

Important is the option „--root ${_builddir}“. With this perl will be installed to our fake-root directory. In the same way you can install further packages which are avaible over pacman.

1 The problem with installing perl modules is that we need to call perl ./Makefile.PL with the option „DESTDIR=$builddir“. But this lead to a path c:\builddir\C:\mingw64\lib\perl5\site_perl… so that dmake install aborts because of wrong path names. I solved this in my script by using INSTALLDIR=VENDOR and overriding INSTALLVENDORBIN, INSTALLVENDORARCH, INSTALLVENDORLIB with relative paths (see details in the install-perl-Gtk3 script!

2 Especially we must not delete all *.exe files because then we loose perl ;-)

3 The MSys2 installer is created by the QT installer framework, too!

- After that we install the perl-Gtk3 modules in the fake root directory. Therefor we have to start the script with the option „-n -d ${_builddir}/“. Please note the conclusive slash at the end!

```
perl /home/maximilian/install-perl-Gtk3-mingw64.pl -n -d ${_builddir}/
```

- Last we copy our application in the „bin“ directory of the MinGW64 environment of our fake-root directory and make it executable with pl2bat:

```
cp /home/maximilian/11_3_toolbar.pl ${_builddir}/mingw64/bin
pl2bat ${_builddir}/mingw64/bin/11_3_toolbar.pl
```

If you want to install your applicatin with the classic, perlish triad *perl ./Makefile.PL*, *dmake*, *dmake install*, you have to pass the attribute „DESTDIR=$builddir“ to perl ./Makefile.PL. But because this leads to a path like c:\$builddir**C:**\mingw64\lib\perl5\site_perl… and *dmake install* therefor aborts because of wrong path names, I get used to pass in addition „INSTALLDIR=VENDOR“ to *perl ./Makefile.PL* and to override the INSTALLVENDOR*-attributes with the correspending relative paths.

For the same reason we have to suppress the appending of installations to perllocal with NO_PERLLOCAL=1.

As a result my perl ./Makefile.PL command would look as follow:

```
perl ./Makefile.PL DESTDIR="$builddir" INSTALLDIRS="vendor" \
INSTALLVENDORARCH="mingw64/lib/perl5/vendor_perl" \
INSTALLVENDORBIN="mingw64/bin" INSTALLVENDORSCRIPT="mingw64/bin" \
INSTALLVENDORLIB="mingw64/lib/perl5/vendor_perl" NO_PERLLOCAL=1
```

Already done!

(3)

Now we have to adjust the QT Installer Framework files:

(a)

Let‘s begin with *qt-ifw/config/config.xml*. The necessary adjustments are almost self-explaining. You have to change the following tags:

```
<Name> … </Name>          # Name of your application being installed

<Title> … </Title>        # Name of the installer as displayed on the title bar

<Publisher> … </Publisher># Publisher of the software (as shown in the Win Control Panel)

<ProductUrl></ProductUrl> # URL to a page that contains product informations on your web site

<StartMenuDir></StartMenuDir> # Name of the default program group of the product
```

The <Version>-Tag will be overrided automatically by the *mak-installer* bash script.

Unfortunately up to now I could not solve, how <RunProgram> can be configured with a perl-Gtk3 application. The problem is, that we have to add „@TargetDir\\bin“ to the PATH environment variable or to run our application from „@TargetDir\\bin“ as working directory, to start our perl program (see below the chapter about creating the shortcut). Alternatively you can show a Readme.txt or ReleaseNote.txt after the installation[4] or just leave <RunProgram> as we want to do it here.

For further informations (e.g. how to show license informations at the beginning of the installer) please study the docs of the QT Installer Framework.

(b)

In the files *qt-ifw/packages/org.MyApp.root/meta/package.xml* and *qt-ifw/packages/org.MyApp..root.base/meta/package.xml*[5] you should change the content of the <DisplayName>- and <Description>-tags. Normally the installer gives the user the opportunity to select which package he wants to install. But because we have deactivated this feature in *qt-ifw/packages/org.MyApp.root/meta/installscript.js* (see the Option "installer.setDefault PageVisible(QInstaller.ComponentSelection, false);"), these variables ultimately have no effect.

(c)

In the file *org.MyApp.root/installscript.js* we have to set the predefined target directory with the following option

```
installer.setValue("TargetDir", systemDrive + "\\MyApp64");
```

Please note that by default the target paths in the QT Installer Framework must not have spaces. You can theoretically work around this limitation by adding the <AllowSpaceInPath>true </AllowSpaceInPath> to your projects *config.xml*-file.[6] But I don‘t recommend this!!! The MSys2 installer doesn‘t allow accents, spaces etc. in the installer folder, too. Therefor and because your binaries are created with MSys2, please build your installer in the same way.

That means:

- Look to it, that the predefined target directory must not contain spaces etc.
- Indicate also your users for example at your web site[7] or install instructions, that they must not install the application into a directory with spaces (e.g. „Program Files“)!!![8]

(d)

Last I want to make you aware of the file *qt-ifw/packages/org.MyApp.root.base/meta/installscript.js*. There are some post-installation-operations for Gtk3 defined.

For us important is especially the creation of a shortcut. Plese note that your binaries are located after the installation in @TARGET_DIR@/bin/*.bat and not in $builddir/mingw64/bin/*.bat as in the fake root directory. In addition we have to set the PATH environment variable or run our application from „@TargetDir@\\bin“ as working directory, because otherwise perl can not be found.

---

4 See the example at http://doc.qt.io/qtinstallerframework/qt-installer-framework-openreadme-example.html
5 Don‘t change the filenames because they are hardcoded in the *make-installer* bash script
6 See https://bugreports.qt.io/browse/QTIFW-148
7 On the MSys homepage you find for example the following instruction: „Enter Installation Folder (ASCII, no accents, spaces nor symlinks, short path)“
8 Alternative to that you can add <AllowSpaceInPath>**false**</AllowSpaceInPath> to config.xml so that the installer prevents spaces in the installation folder.

The following examples worked for me:

```
# Setting the PATH environment variable at startup
var cmdLocation = windir + "\\system32\\cmd.exe";
component.addOperation("CreateShortcut",
        cmdLocation,
        "@StartMenuDir@/11_3_toolbar.lnk",
        "/A /Q /K SET PATH=@TargetDir@\\bin;%PATH% & MyApp.bat");

# If you don‘t want to use pl2bat, you can try also
# Note that you have to pass to perl the absolute path to your perl scipt (I don‘t know why...)
var cmdLocation = windir + "\\system32\\cmd.exe";
component.addOperation("CreateShortcut",
        cmdLocation,
        "@StartMenuDir@/11_3_toolbar.lnk",
        /A /Q /K SET PATH=@TargetDir\\\bin;%PATH% & perl @TargetDir\\bin\\MyApp.pl

# Starting from @TargetDir\\bin as working directory works, too
var cmdLocation = windir + "\\system32\\cmd.exe";
component.addOperation("CreateShortcut",
        cmdLocation,
        "@StartMenuDir@/11_3_toolbar.lnk",
        "/A /Q /K @TargetDir@\\MyApp.bat",
        „workingDirectory=@TargetDir\\bin“);

# If you want to close the cmd shell automatically after your app is closed,
# then you have to use the cmd - option /C instead of /K
var cmdLocation = windir + "\\system32\\cmd.exe";
component.addOperation("CreateShortcut",
        cmdLocation,
        "@StartMenuDir@/11_3_toolbar.lnk",
        "/A /Q /C SET PATH=@TargetDir@\\bin;%PATH% & MyApp.bat");
```

For further informations please read the documentation of the QT Installer Framework.

(4)

After the adjustment of the files the creation of the installer is completed automaticly. Just start the Batch file

```
./make-installer.bat
```

in your MinGW64 shell.

4) Conclusion

I hope this HowTo was helpful for you. If you find any programming mistakes, misspellings or other mistakes, please let me know (Maximilian-Lika@gmx.de). I am very grateful for every improvement suggestions!

If you have further questions regarding using perl-Gtk3 under windows the best place to ask is the GTK-Perl mailing list (gtk-perl-list@gnome.org).