Perl-Gtk3 und Windows

Ein kleines HowTo, wie man unter Windows perl-Gtk3 Programme und einen Windows Installer erstellt

1) Installation der MSys2 Umgebung

Zur Erstellung eines Windows Installers für eine perl-Gtk3 Anwendung müssen Sie zunächst die MSYS2 Umgebung installieren.

Laden Sie hierfür unter www.msys2.org die x86_64-Installationsdatei herunter und befolgen Sie die Installationsanweisungen auf der Homepage. Vergessen Sie insbesondere nicht die Paket-Datenbank und das MSys2-Kernsystem upzudaten (mehr Informationen unter www.msys2.org)

2) Installation des Gtk3 Perl Moduls

Daraufhin müssen Sie in Ihrer MinGW64 Umgebung die perl-Gtk3 Module installieren. Dies ist zwar etwas umständlich, da wir bei der Erstellung des Windows Installers diese erneut unter einem alternativen Root-Verzeichnis installieren werden. Allerdings macht es die Sache etwas leichter. Zu dem erhalten Sie auf diese Weise eine wunderbare Testumgebung für Ihre perl-Gtk3 Programme.

a)

Zunächst installieren Sie Perl mit folgendem Befehl:

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

b)

Die darüberhinaus benötigten Bibliotheken und Module, können Sie bequem über das perl-Gtk3 Installationsskript installieren:

Laden Sie hierzu das Perl Skript herunter unter https://github.com/MaxPerl/perl-Gtk3-windows-installer und installieren Sie das perl-Gtk3 Modul mittels

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

3) Erstellung des Windows Installers

Nachdem alles erfolgreich installiert wurde, können Sie nun für Ihr Programm einen Windows-Installer erstellen. Wir demonstrieren die Erstellung hier mittels eines kleinen perl-Gtk3-Beispielskripts aus meinem perl Gtk3 Tutorial (https://github.com/MaxPerl/perl-Gtk3-Tutorial). Das Programm besteht also de facto aus nur einer einzigen Datei, die wir einfach dadurch “installieren”, dass wir sie an die richtige Stelle kopieren und mittels pl2bat ausführbar machen.

Aber auch komplexere Anwendungen dürften kein Problem sein. Wenn Sie bspw. die Anwendung mit ExtUtils::MakeMaker gepackt haben, können Sie dieses ganz einfach auch dadurch installieren, dass Sie das install-perl-Gtk3-mingw64.pl Skript nach der Zeile "$exitcode = install_perl_module('Gtk3');" um folgenden Eintrag

ergänzen (noch nicht implemtiert!!! Nur Module auf CPAN funktionieren derzeit):[1]

install_perl_module('MyApp::Module');

Nichtsdestotrotz wollen wir hier aus Vereinfachungsgründen von nur einer aus einer einzelnen Datei bestehenden Anwendung ausgehen, welche einfach dadurch "installiert" wird, dass wir sie in das richtige Verzeichnis kopieren und daraus mittels pl2bat eine ausführbare .bat Datei erstellen.

a)

Der Installer wird mittels eines Bash-Skriptes und mit Hilfe des QT Installer Framework erstellt. Sie können alle hier verwendeten Dateien unter https://github.com/MaxPerl/perl-Gtk3-Tutorial/tree/master/create_windows_installers_template herunterladen. Die Dateien stammen ursprünglich vom dem gedit Windows Installer Projekt (siehe https://blogs.gnome.org/nacho/2014/08/01/how-to-build-your-gtk-application-on-windows/ und https://git.gnome.org/browse/gedit/tree/win32). Ich habe sie nur an wenigen Stellen leicht modifiziert.[2]

Das Gedit Team ist zwischenzeitlich zu WiX gewechselt. Nichtsdestotrotz wollen wir hier auch weiterhin das QT Installer Framework nutzen, da ich dieses übersichtlicher finde und v.a. auch in MSys2 enthalten ist.[3]

Zur Erstellung des Windows Installers sind nur wenige Anpassungen notwendig:

(1)

Im *make-installer* Bash Skript sollten Sie die Variablen am Beginn anpassen:

```
_filename       # Der Dateinamen des zu erstellenden Windows Installers

_version        # Die Version Ihrer Anwendung

_builddir       # Das „Fake-Root"-Verzeichnis, in dem die Anwendung  gebaut wird, bevor Sie zu
                dem Installer zusammengepackt wird

_perlgtkinstallskript   # Der Standort des perl-Gtk3 Installationsskript (s.o.)
```

(2)

Als nächstes sollten Sie die Funktion *install_application_packages()* so anpassen, dass alle zur Installation erforderlichen Bibliotheken, Module und Dateien in dem Fake-Root Verzeichnis installiert werden. In unserem Beispiel sind hierzu folgende Einträge erforderlich:

- Zunächst installieren wir Perl mittels

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

Wichtig ist hier die Option --root "${_builddir}". Damit werden die Pakete in dem

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!

Fake-Root Verzeichnis installiert. Auf dieselbe Weise können Sie problemlos weitere Pakete installieren, die über pacman erhältlich sind und zum Laufen Ihres Programms erforderlich sind

- Als Nächstes installieren wir die Perl-Gtk3 Module in das Fake-Root Verzeichnis. Hierfür müssen wir das Perl-Gtk3 Installationsskript mit der Option "-d ${_builddir}/" aufrufen. Beachten Sie insbesondere den abschließenden Schrägstrich!

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

- Zuletzt kopieren wir unsere Anwendung in das bin-Verzeichnis der MinGW64 Umgebung unseres Fake-Roots-Verzeichnis und machen dieses mittels *pl2bat* ausführbar:

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

Wenn Sie Ihr Programm mit dem klassischen, perlischen Dreiklang *perl ./Makefile.PL*, *dmake*, *dmake install* installieren wollen, müssen Sie perl ./Makefile.PL mit der Option „DESTDIR=$builddir“ aufrufen. Da allein diese Option komischerweise dazu führt, dass perl als Installationsverzeichnis c:\$builddir**C:**\mingw64\lib\perl5\site_perl… annimmt und die Installation daher scheitern würde, habe ich mir angewohnt zusätzlich „INSTALLDIR=VENDOR“ auszuwählen und die entsprechenden INSTALLVENDOR*-Attribute mit den korrespondierenden relativen Pfadangaben zu überschreiben.

Aus demselben Grund müssen wir auch das Hinzufügen der Installationen zu perllocal mit dem Attribut NO_PERLLOCAL=1 verhindern.

Im Ergebnis würde das perl ./Makefile Kommando bei mir folgendermaßen aussehen:

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

Schon ist das make-installer Skript fertig angepasst!

(3)

Als nächstes müssen wir die QT Installer Framework Dateien anpassen:

(a)

Beginnen wir am besten mit der Datei *qt-ifw/config/config.xml* an. Die erforderlichen Anpassungen sind überwiegend selbsterklärend. Wir müssen insbesondere folgende Tags ändern:

```
<Name> … </Name>          # Name der zu installierenden Anwendung

<Title> … </Title>        # Name des Installers, wie er in Titelleiste angezeigt werden soll

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

<ProductUrl></ProductUrl> # URL zu der Produkt-Webseite
```

```
<StartMenuDir></StartMenuDir> # Name der voreingestellten Programmgruppe
```

Der <Version>-Tag wird automatisch durch das *make-installer* bash Skript überschrieben und angepasst.

Bislang ist es mir leider nicht gelungen, <RunProgram> mit einer perl-Gtk3 Anwendung erfolgreich zu konifgurieren. Das Problem ist, dass wir zum Starten einer perl-Gtk3 Anwendung zunächst die PATH Systemvariable um "@TargetDir\\bin" erweitern müssen oder "@TargetDir\\bin" als Arbeitsverzeichnis einstellen müssen (siehe Details unten im Kapitel über die Erstellung der Startverknüpfung). Alternativ können Sie nach der Installation natürlich eine Textdatei (z.B. ReadMe.txt oder ReleaseNotes.txt) anzeigen lassen[4] oder wie wir hier diese Einstellung einfach weglassen.

Für weitere Informationen (bspw. wie Sie in Ihrem Installer einen Lizenztext anzeigen lassen können), lesen Sie bitte in der Dokumentation zum QT Installer Framework nach.

(b)

In den Dateien *qt-ifw/packages/org.MyApp.root/meta/package.xml* sowie *qt-ifw/packages/org.MyApp.root.base/meta/package.xml*[5] sollten Sie jeweils den Inhalt des <DisplayName>- und des <Desciption>-Tags anpassen. Normalerweise könnte man dem Anwender eine Auswahlmöglichkeit geben, welche packages er installieren will. Da wir dieses Feature jedoch in der Datei *qt-ifw/packages/org.gnome.gedit.package/meta/installscript.js* (siehe die Option"installer.setDefault PageVisible(QInstaller.ComponentSelection, false);"), haben diese Variablen im Ergebnis keine Bedeutung.

(c)

In der Datei *org.MyApp.root/installscript.js* müssen wir das voreingestellte Zielverzeichnis mit der folgenden Option einstellen:

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

Bitte beachten Sie, dass das Zielverzeichnis bei dem QT Installer Framework standardmäßig keine Leerzeichen enthalten darf. Theoretisch kann man diese Einschränkung dadurch umgehen, dass man der *config.xml*-Datei <AllowSpacing>true</AllowSpacing> hinzufügt.[6] Ich rate hiervon jedoch dringend ab!!! Der MSys2 Installer erlaubt selbst keine Akzente, Leerzeichen etc. im Installationsordner. Daher und weil ja unsere ausführbaren Dateien unter MSys2 erstellt wurden, erstellen Sie Ihren Installer auf demselben Weg wie es das MSys2 Team selbst macht!

Das bedeutet im Ergebnis zweierlei:

- Achten Sie darauf, dass das voreingestellte Zielverzeichnis keine Leerzeichen enthält
- Weisen Sie v.a. auch Ihre Benutzer bspw. auf Ihrer Homepage[7] oder in einer Installationsanleitung darauf hin, dass Sie die Anwendung nicht in ein Verzeichnis installieren dürfen, dass Leerzeichen enthält (also auch nicht in "Programm Files"!!!)[8]

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

(d)

Zuletzt möchte ich Sie noch auf die Datei *qt-ifw/packages/org.gnome.gedit.root.package/meta/installscript.js* aufmerksam machen. Hier werden einige Post-Installations-Operatione für Gtk3 definiert.

Bedeutend ist für uns v.a. die Erstellung eines Shortcuts. Beachten Sie bitte, dass Ihre Programme nach der Installation in dem Verzeichnis @TARGET_DIR@/bin/*.bat sein werden, und nicht wie im Fake-Root Verzeichnis unter $builddir/mingw64/bin/*.bat. Außerdem müssen wir die PATH Systemvarbiable erweitern bzw. “@TargetDir@\\bin” als Arbeitsverzeichnis einstellen, weil sonst perl nicht gefunden werden kann.

Die folgenden Beispiele funktionierten für mich:

```
# Beim Programaufruf jeweils die PATH Variable setzen
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");

# Wenn Sie pl2bat nicht nutzen wollen, können Sie auch folgendes probieren
# Achten Sie nur darauf, dass Sie Perl dem absoluten Pfad zu Ihrem Perl Skript übergeben müssen
# (Ich weiß auch nicht, warum...)
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

# Das Starten aus dem Arbeitsverzeichnis @TargetDir\\bin funktionierte bei mir auch
var cmdLocation = windir + "\\system32\\cmd.exe";
component.addOperation("CreateShortcut",
        cmdLocation,
        "@StartMenuDir@/11_3_toolbar.lnk",
        "/A /Q /K @TargetDir@\\MyApp.bat",
        „workingDirectory=@TargetDir\\bin“);

# Wenn Sie wollen, dass die cmd Shell automatisch beendet wird, wenn Ihre Anwendung
# beendet wird, müssen Sie die cmd-Option /C anstatt /K verwenden:
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");
```

Für Details darf ich Sie auf die Dokumentation des QT Installations Framework hinweisen.

(4)

Die Erstellung des Installers geht nun, nachdem Sie alles auf Ihre Bedürfnisse angepasst haben, wieder automatisch, in dem Sie in Ihrer MinGW64 Shell einfach das Batch Skript

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

prevents spaces in the installation folder.

ausführen.

4) Conclusion

Ich hoffe, dieses einfache HowTo war Ihnen eine kleine Hilfe. Wenn Sie irgendwelche Programmierfehler, Rechtschreibfehler oder sonstige Fehler finden, können Sie mir gerne eine eMail schreiben (Maximilian-Lika@gmx.de). Auch darüber hinaus bin ich für jegliche Verbesserungsvorschläge bzgl. dieses Tutorials bzw. der verwendeten Skripte dankbar.

Falls Sie weitere Fragen bzgl. Perl-Gtk3 und Windows haben, ist der beste Ort die Gtk-Perl Mailing-Liste (gtk-perl-list@gnome.org)