Lazarus 1.6

Installieren und fürs Windows-Cross-Compiling einrichten

Autor : Heiko Rompel

Version 1.3

Einleitung

In dieser Anleitung wird erklärt welche Schritte nötig sind um unter ***Linux Mint 17.3*** ***d***ie Entwicklungsoberfläche ***Lazarus 1.6*** zu installieren.

Des weiteren wird beschrieben wie man ***Lazarus 1.6*** und ***Freepascal 3.0.0*** einrichten muss, um damit auch Programme für ***Windows 32-Bit*** und ***Windows 64-Bit*** Oberflächen zu erstellen.

Ohne einen Artikel im ***Blaise Pascal Magazin 10/2015***
http://www.blaisepascal.eu/ ,

der Hilfe der Mitglieder im **Freepascal-Forum**
http://forum.lazarus.freepascal.org/index.php/topic,31515.0.html

und dem deutschen ***Lazarus-Forum***
http://www.lazarusforum.de/viewtopic.php?f=16&t=9411
hätte ich es nie geschafft.

Ich hoffe, Ihr könnt den Anweisungen folgen und kommt damit auch ans Ziel.

Vielleicht, liefert ja mal jemand auf Basis dieser Anleitung eine Anleitung um ***Android***-Apps, **iOS**-Programme und so weiter zu erstellen.

Bremerhaven, den 21.02.2016

Inhaltsverzeichnis

Lazarus installieren

Ich gehe davon aus, dass ***Linux Mint 17.3 schon fertig installiert ist.***

Als nächstes müssen Ihr also Lazarus installieren.

Ihr nutzt hierfür das nachfolgendes Script aus dem ***Blaise Pascal Magazin 10/2015.***
(Das Script wurde schon auf Lazarus 1.6 (ohne RC) umgestellt.)

Markiert Euch das Script und fügt es in eine Textdatei ein.
Benennt die Datei in getlaz.sh.

Ihr startet das Script in einem Terminal-Fenster mit dem Befehl: **sudo sh getlaz.sh**

Das Script für Lazarus

```
#!/bin/bash
#
# run in your HOME-directory with: sudo sh getlaz.sh  (HR)
#
# (C)by: Michael Van Canneyt (michael@freepascal.org)
#
# Release from: 2016-01-30  (HR)
# additional comments from: Heiko Rompel (HR)
##############################################################
# Some variables. Set this to whatever you want
##############################################################
#
# Where to download/install everything ? (below home directory)
#
INSTALLDIR=fpc-install
#
# Which FPC version to use ?
#
VERSION=3.0.0
CPUARCH=`uname -p`
#
# Install FPC/Lazarus as root ? (YES or NO)
#
USEROOT=YES
#
# Which lazarus version ?
# set either tag or branch variable.
# If neither is set, trunk is used.
# When lazarus 1.6 is out, this becomes lazarus_1_6
```

```
#
# A newer RC is out (HR)
TAG=lazarus_1_6/
BRANCH=

############################################################
# No variables after this point.
############################################################
#
# Install preliminaries. This must be done as root.
#
sudo apt-get install subversion make binutils gdb gcc libgtk2.0-dev

#
# Check if the rest must be done as root.
#
if [ "$USEROOT" = YES ]; then
  SUDO=sudo
fi
#
############################################################
############
# Get and install FPC.
############################################################
############
#
# Create installation directory
#
mkdir ~/$INSTALLDIR
cd ~/$INSTALLDIR
#
# Fetch the necessary files for FPC.
#
wget ftp://ftpmaster.freepascal.org/pub/fpc/dist/$VERSION/source/fpc-
$VERSION.source.tar.gz
wget ftp://ftpmaster.freepascal.org/pub/fpc/dist/$VERSION/$CPUARCH-
linux/fpc-$VERSION.$CPUARCH-linux.tar
#
# extract installer.
#
tar xvf fpc-3.0.0.$CPUARCH-linux.tar
cd fpc-3.0.0.$CPUARCH-linux
#
# Install FPC (possibly as root)
#
$SUDO sh ./install.sh
```

```
#
# Extract sources.
#
cd ~
tar xvzf $INSTALLDIR/fpc-$VERSION.source.tar.gz
cd ~/$INSTALLDIR
#
#############################################################
#############
# Get and install Lazarus
#############################################################
#############
#
# Determine SVN url
#
BASEURL=http://svn.freepascal.org/svn/lazarus/
#
if [ ! -z "$TAG" ]; then
  SVNURL=$BASEURL/tags/$TAG
else
  if [ ! -z "$BRANCH" ]; then
    SVNURL=$BASEURL/branches/$BRANCH
  else
    SVNURL=$BASEURL/trunk
  fi
fi
#
# Check out sources
#
svn co $SVNURL lazarus
#
# Build the IDE
#
cd lazarus
make bigide
#
# Install lazarus (possibly as root)
#
$SUDO make install
#
# That's all folks !
#
```

Nach dem Script

Auch wenn es ungewöhnlich für ***Linux*** ist, meldet Euch an dieser Stelle einmal ab und wieder an.
(Bei mir wurde sonst nicht die Programmgruppe „Entwicklung“ im Hauptmenü angelegt.)

Crosscompiling für Windows 32-Bit einrichten

So jetzt habt Ihr schon ein lauffähiges ***Lazarus*** mit dem Ihr unter ***Linux*** Programme erstellen könnt. Aber, Ihr wollt ja mehr.

Jetzt kommen die Informationen aus dem ***Lazarus-Wiki***
http://wiki.lazarus.freepascal.org/Cross_compiling_for_Win32_under_Linux zum Einsatz.

Nachfolgende Zeilen werden nacheinander in einem Terminal-Fenster eingegeben, das in diesem Verzeichnis „/home/**laz-user**/fpc-3.0.0“
(**laz-user** gegen Deinen Usernamen ersetzen) gestartet wird :

$ sudo make all OS_TARGET=win32 CPU_TARGET=i386

und dann

$ su -c "make crossinstall OS_TARGET=win32 CPU_TARGET=i386"

Auch wenn man laut Wiki damit schon fertig ist, fehlen noch ein paar Anweisungen.

In der Datei „/etc/fpc.cfg „ muss noch diese Zeile eingefügt werden:

-Fu/usr/local/lib/fpc/$fpcversion/units/$fpctarget/*

Die Datei ist zumindest bei mir schreibgeschützt und kann nur in einem Terminal-Fenster geöffnet werden, das mittels „Als Systemverwalter öffnen“ geöffnet wurde.

Jetzt kommt noch ein symbolischer Link:

$ sudo ln -s /usr/local/lib/fpc/3.0.0/ppcross386 /usr/bin/

Das „$“-Zeichen braucht Ihr nicht mit eingeben.

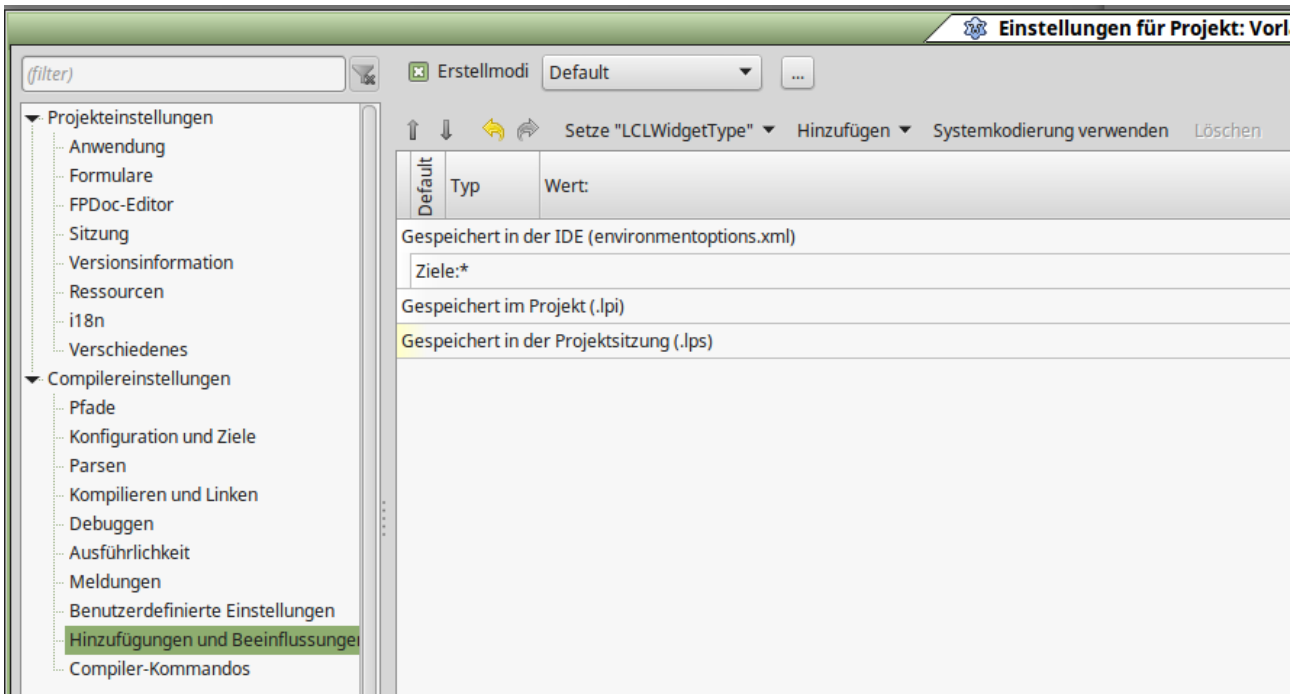
Lazarus für Windows 32-Bit einrichten

Jetzt geht es mit der Einrichtung in Lazarus weiter.
(Hier werdet Ihr teilweise auch gleich die Einrichtung für die 64-Bit-Version mit machen.)

Die nachfolgenden Einstellungen finden statt unter „Projekt / Projekteinstellungen“.

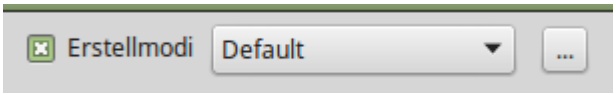
1. Hinzufügungen und Beeinflussungen

Wenn man diesen Punkt das erste Mal aufruft, dann sieht der so aus:



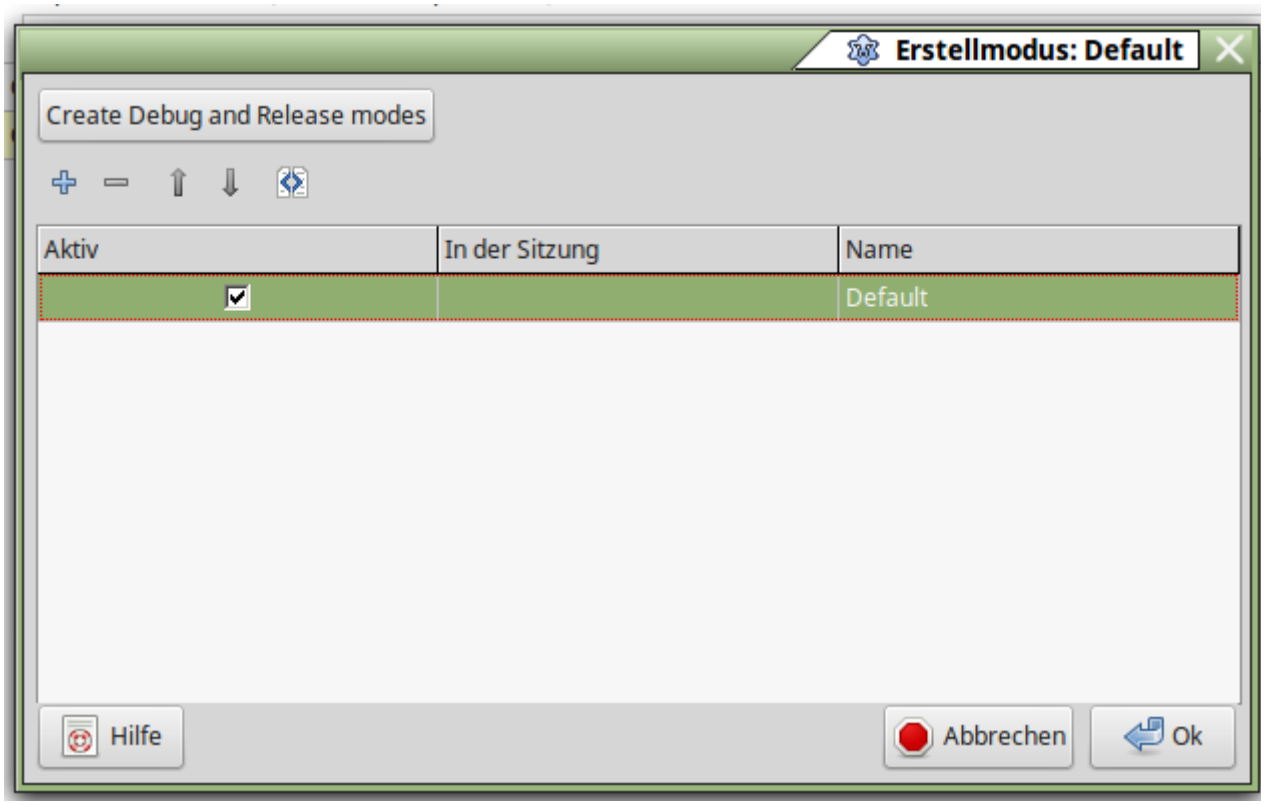
Die Erstellmodi

Als erstes legt Ihr jetzt die „Erstellmodi“ an.



Bitte auf die drei Punkte klicken.

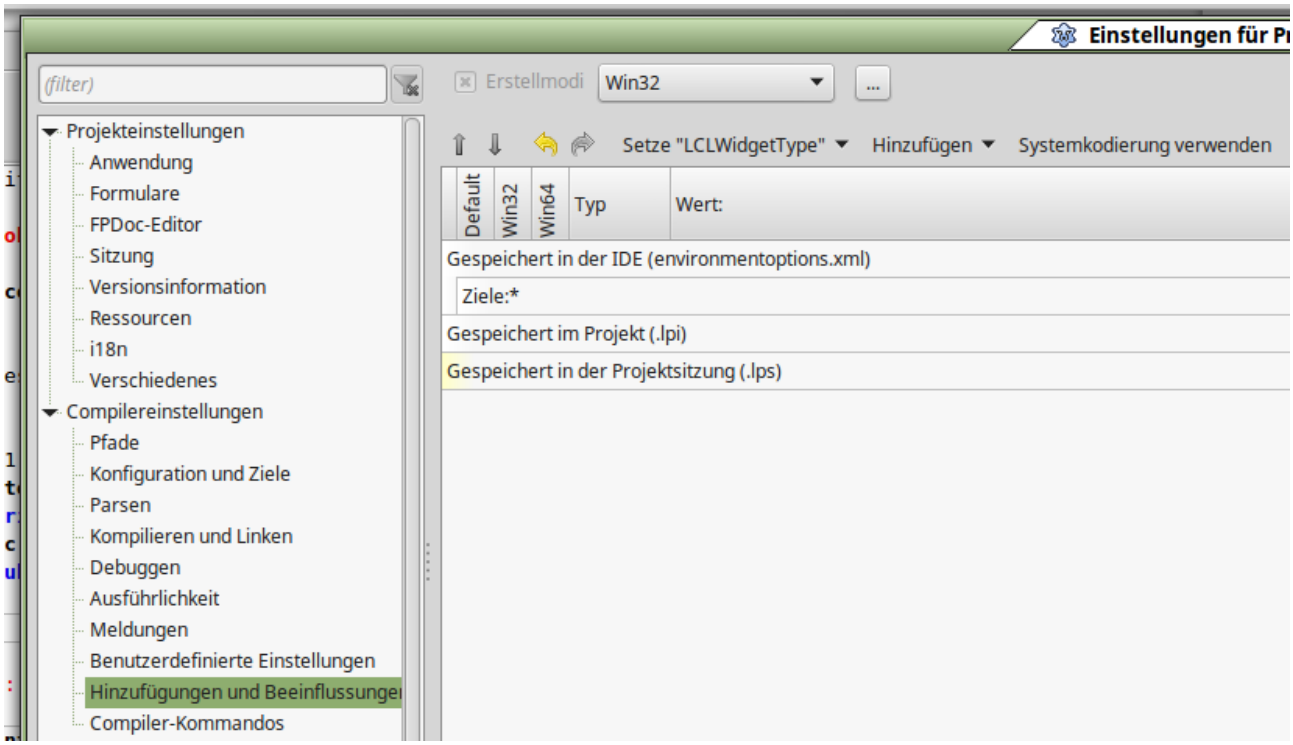
Er erscheint dieser Dialog:



Hier klicken Ihr jetzt auf das „+“-Zeichen.

Es erscheint eine neue Zeile.
In der Spalte „Name“ tragt Ihr **„Win32“** ein.
Ihr erstellt noch eine neue Zeile und benennen sie **„Win64“**.
Jetzt habt Ihr drei „Erstellmodi“ zur Auswahl.
Default: Hier sind die Werte so wie direkt nach der Installation und in diesem Fall für das erstellen von Linux-Programmen.
Win32 und Win64 sind selbsterklärend und werden nachfolgend mit weiteren Werten eingerichtet.

Unser Einstellungsdialog sieht jetzt so aus:



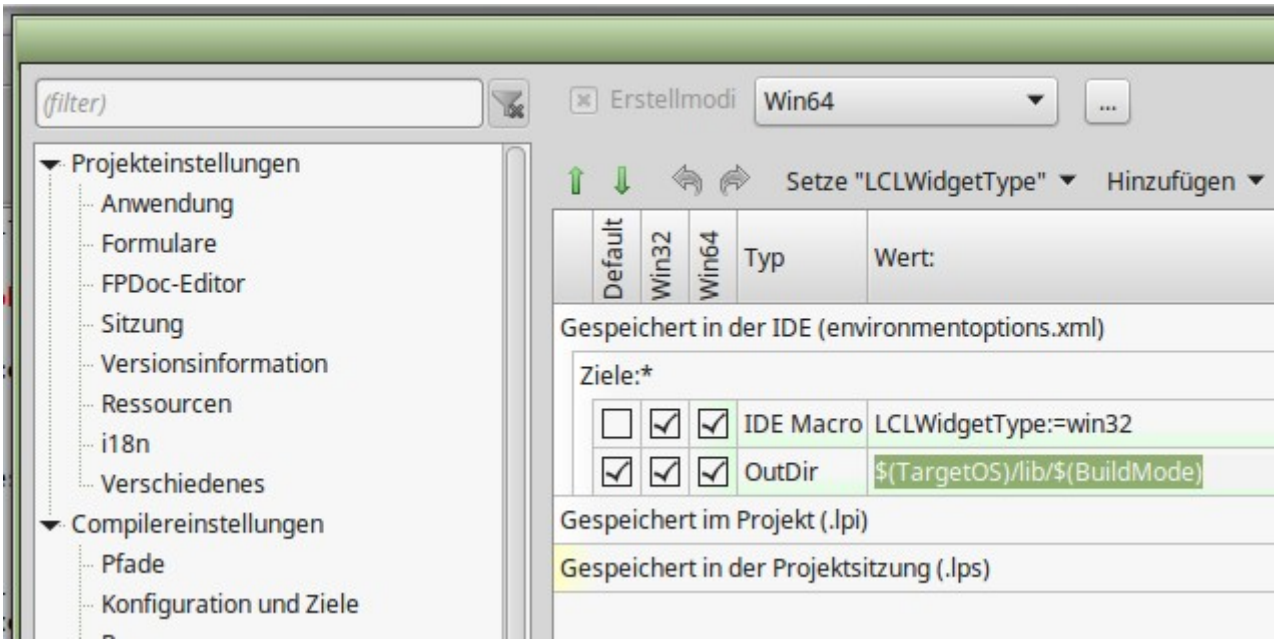
Jetzt müsst Ihr erst der „Erstellmodi“ „Win32“ auswählen.

Die weiteren Einstellungen wollen wir in der IDE speichern.
Ihr klickt deshalb in die Zeile „Gespeichert in der IDE ...“.

Nun wählt Ihr aus der Liste „Setze „LCLWidgetType““ den Wert „Win32“ aus.
Wenn Ihr den Wert ausgewählt habt, setzt Ihr in der Zeile je einen Haken bei „Win32“ und „Win64“. Das bedeutet, dass dieser Wert für beide „ErstellModi“ gilt.

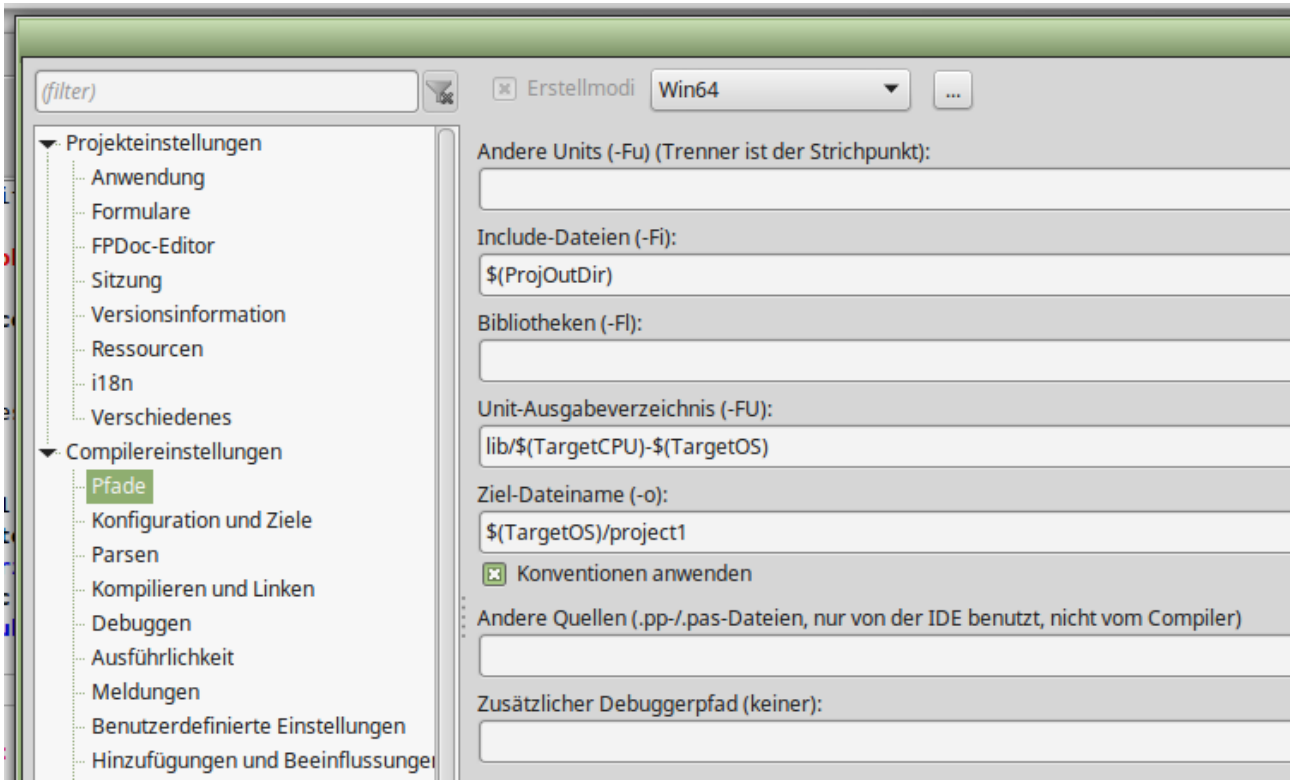
Damit Ihr nachher im Projektverzeichnis etwas mehr Übersicht habt, wählt Ihr jetzt noch aus der Liste „Hinzufügen „ den Punkt „Ausgabeverzeichnis ersetzen“ und füllt die Spalte „Wert“ mit : $(TargetOS)/lib/$(BuildMode)
So wird für jedes Zielbetriebssystem ein Unterverzeichnis erstellt.

Eurer Dialog sollte jetzt so aussehen:



Als nächstes wendet Ihr Euch den Pfad-Einstellungen zu.

2. Pfade



Hier wählt Ihr oben nacheinander alle „Erstellmodi“ aus und tragt in die Zeile „Ziel-Dateiname (-o)“ folgendes ein:

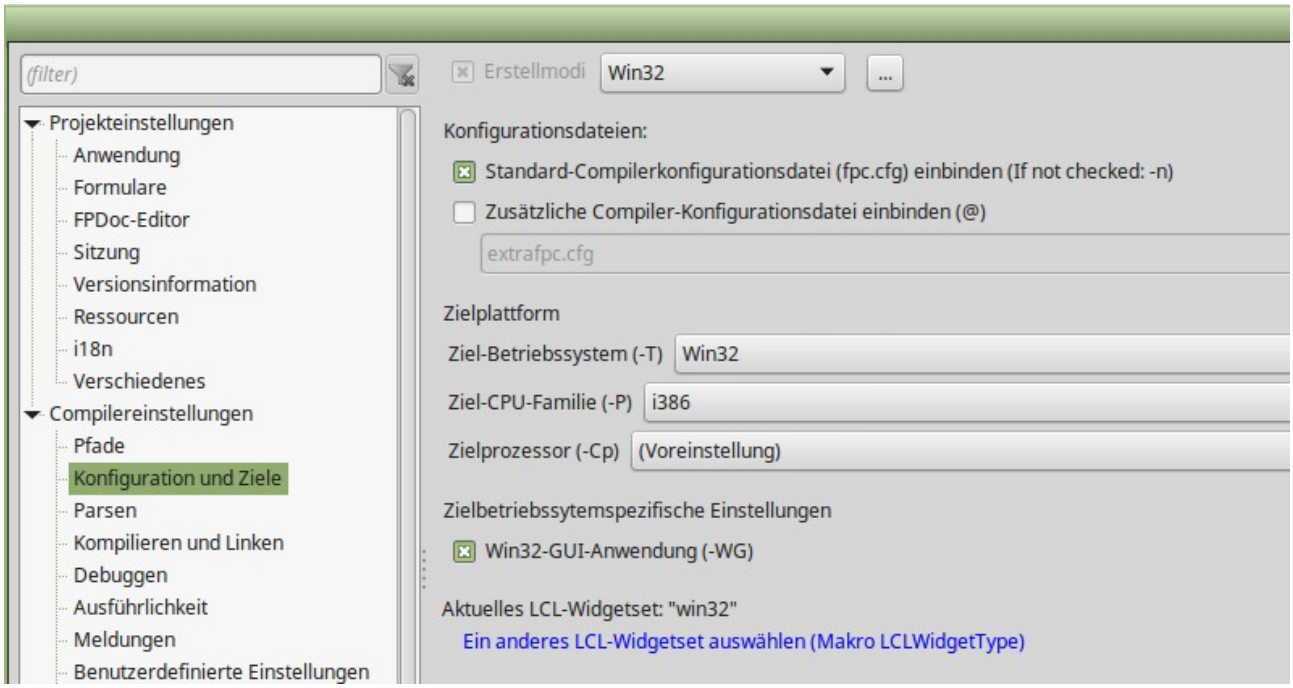
$(TargetOS)/project1

Hierdurch werden auch die erstellten, ausführbaren Dateien im Projektverzeichnis in entsprechende Unterverzeichnisse einsortiert.

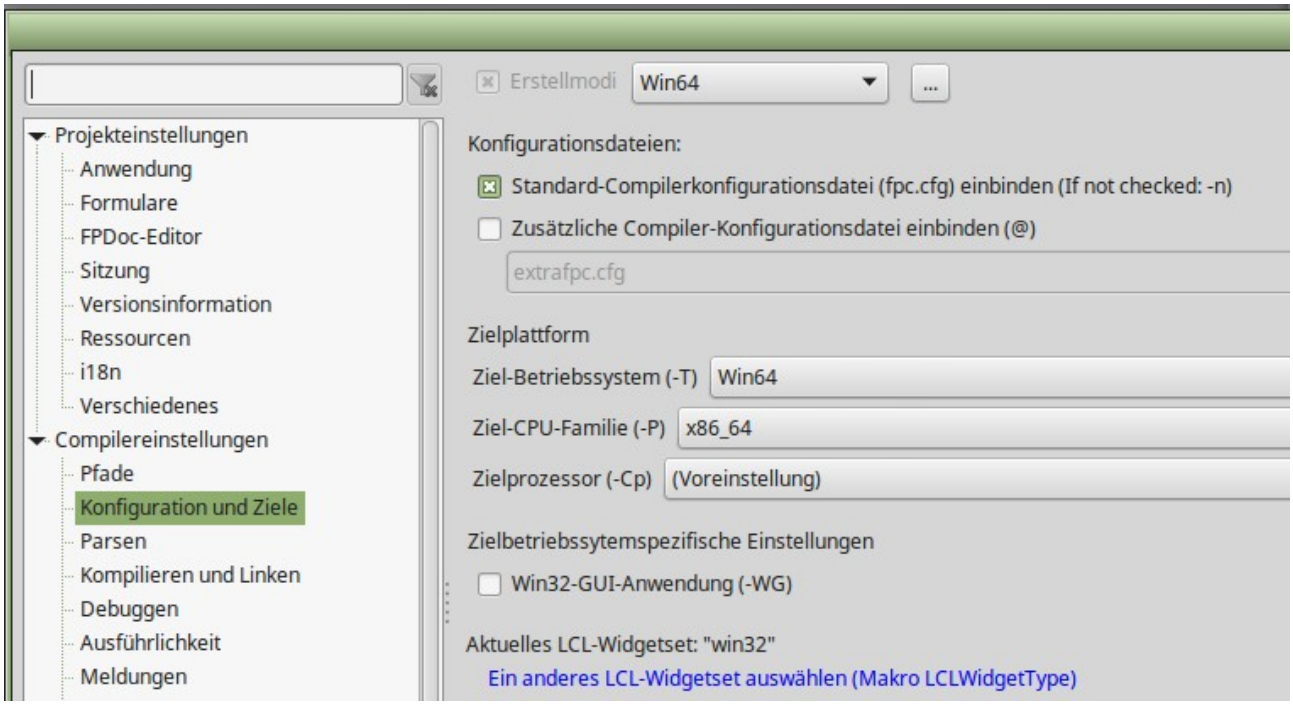
3. Konfiguration und Ziele

Diese Einstellungen sind nötig, wenn Ihr auf einem 32-Bit-System für ein 64-Bit-Sytem oder umgekehrt, entwickelt.

Für Win32-Anwendungen



Für Win64-Anwendungen



Wenn Ihr alles richtig gemacht habt, könnt Ihr jetzt schon Programme für 32-Bit ***Windows-*** Systeme erstellen.

Die ***Windows*** Programme die Ihr unter ***Linux*** erstellt, werden natürlich nicht nach dem Erstellen in der ***Linux***-Maschine ausgeführt.

Um ***Windows***-Programme direkt unter Linux laufen zu lassen, braucht Ihr entweder ***WINE*** oder ein echtes ***Windows*** in einer virtuellen Maschine.

So jetzt zu der Win64-Konfiguration.

Crosscompiling für Windows 64-Bit einrichten

Jetzt kommen die zusätzlichen Informationen aus dem ***Lazarus-Wiki*** http://wiki.lazarus.freepascal.org/Cross_compiling_for_Win32_under_Linux zu Einsatz.

Nachfolgende Zeilen werden nacheinander in einem Terminal-Fenster eingegeben, das in diesem Verzeichnis: „/home/**laz-user**/fpc-3.0.0“
(**laz-user** gegen Deinen Usernamen ersetzen) gestartet wird :

$ sudo make all OS_TARGET=win64 CPU_TARGET=x86_64

und dann

$ su -c "make crossinstall OS_TARGET=win64 CPU_TARGET=x86_64"

Jetzt kommt noch ein symbolischer Link:

$ sudo ln -s /usr/local/lib/fpc/3.0.0/ppcrossx64 /usr/local/bin

Das „$“-Zeichen braucht Ihr nicht mit eingeben.

WINE in Lazarus einbinden

Um auch **Windows**-Programme durch F9 starten zu lassen, muss man **WINE** in **Lazarus** einbinden.

Als erstes muss man **WINE** durch die Anwendungsverwaltung von **Linux Mint** installieren lassen.

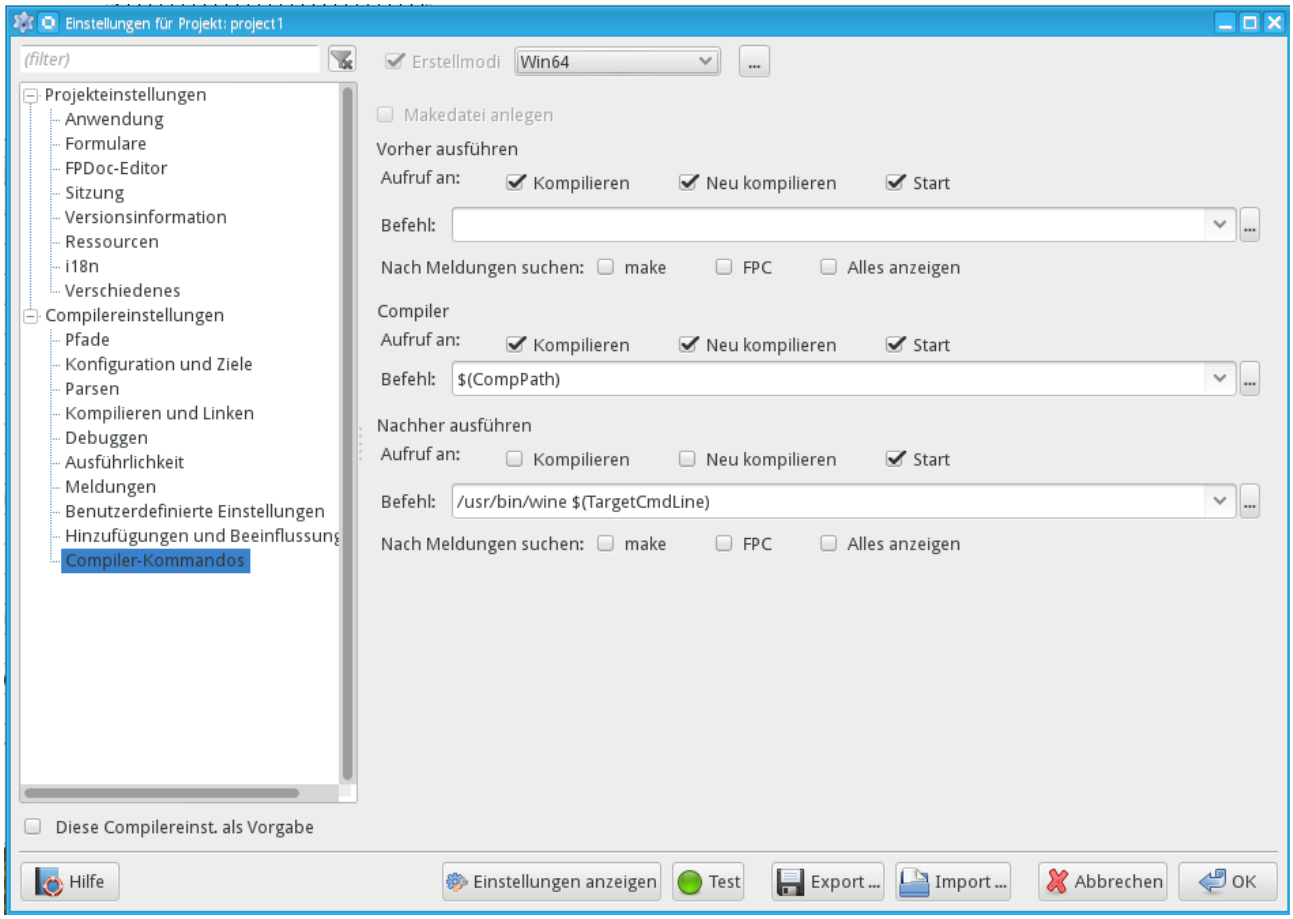
Wenn man das erledigt hat, muss man **Lazarus** noch mitteilen, dass es Win32/64-Programme an **WINE** weiterreichen soll.

Dieses geschieht unter „Projekt / Projekteinstellungen / Compiler-Kommandos“.

Wie schon zuvor kann man hier Einstellungen für jeden definierten Erstellmodus vornehmen.

Für Win32 und Win64 muss man unter „Nachher ausführen: / Befehl“ den Befehl:

„/usr/bin/wine $(TargetCmdLine) eintragen und „[ ] Kompilieren“ und „[ ] Neu kompilieren“ deaktivieren.

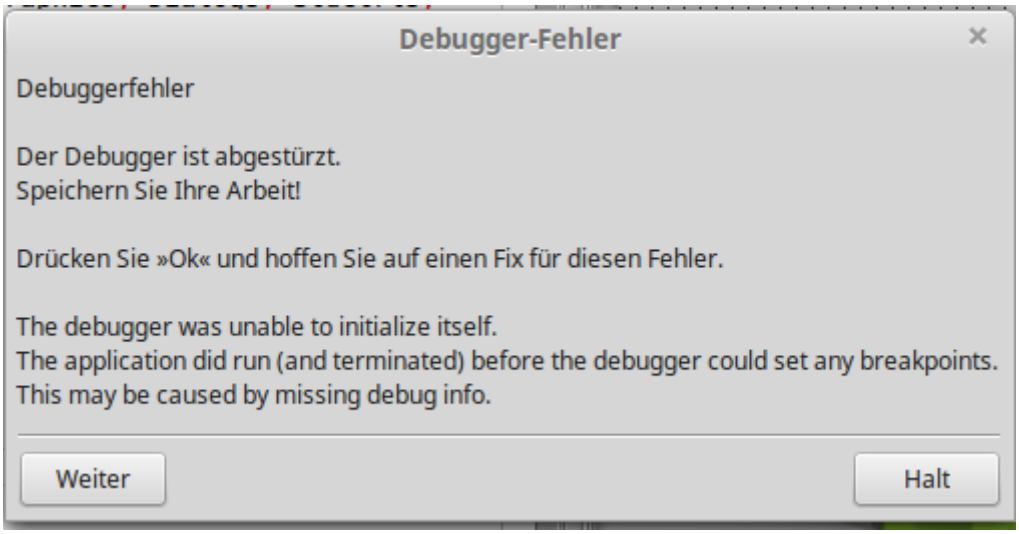


Wenn Ihr die Befehlszeile per Copy&Paste einfügt, passt auf das am Anfang **kein Leerzeichen** steht – sonst funktioniert das alles nicht.

Wenn man dann das erste Mal eine **Windows**-Anwendung mittels F9 startet, teilt **WINE** noch mit, das es „Mono“ und zweimal „Gecko“ braucht und bietet auch an, diese zu installieren.

Jetzt sollten Win32-Anwendungen und Win64-Anwendungen automatisch in **WINE** ausgeführt werden.

Das Einzige was ein wenig stört ist das Fenster, das erscheint, wenn man die Win32/64-Anwendung schliesst:



Einfach auf „Halt“ klicken und Lazarus läuft normal weiter.

Vielleicht hat ja jemand eine Lösung für dieses kleine Problem und teilt sie mir mit.

Gruß Heiko

info@rompelsoft.de

Schlusswort

Ich hoffe, ich kann vielen mit dieser Anleitung weiterhelfen.

Copyright-Hinweise

- Lazarus
- Freepascal
- Windows
- Linux
- Linux-Mint
- Blaise Pascal Magazin
- WINE

Sind Markenbezeichnungen der verschiedenen Hersteller.

Durch die Nutzung dieser Bezeichnungen habe ich keinerlei finanzielle oder sonstige Vorteile.

Danksagung

Ich danke meiner Frau für das Korrekturlesen und meinem Sohn für die englische Übersetzung.