Cross Compiler

Entwickeln von Windows Anwendungen unter Linux (und mehr)

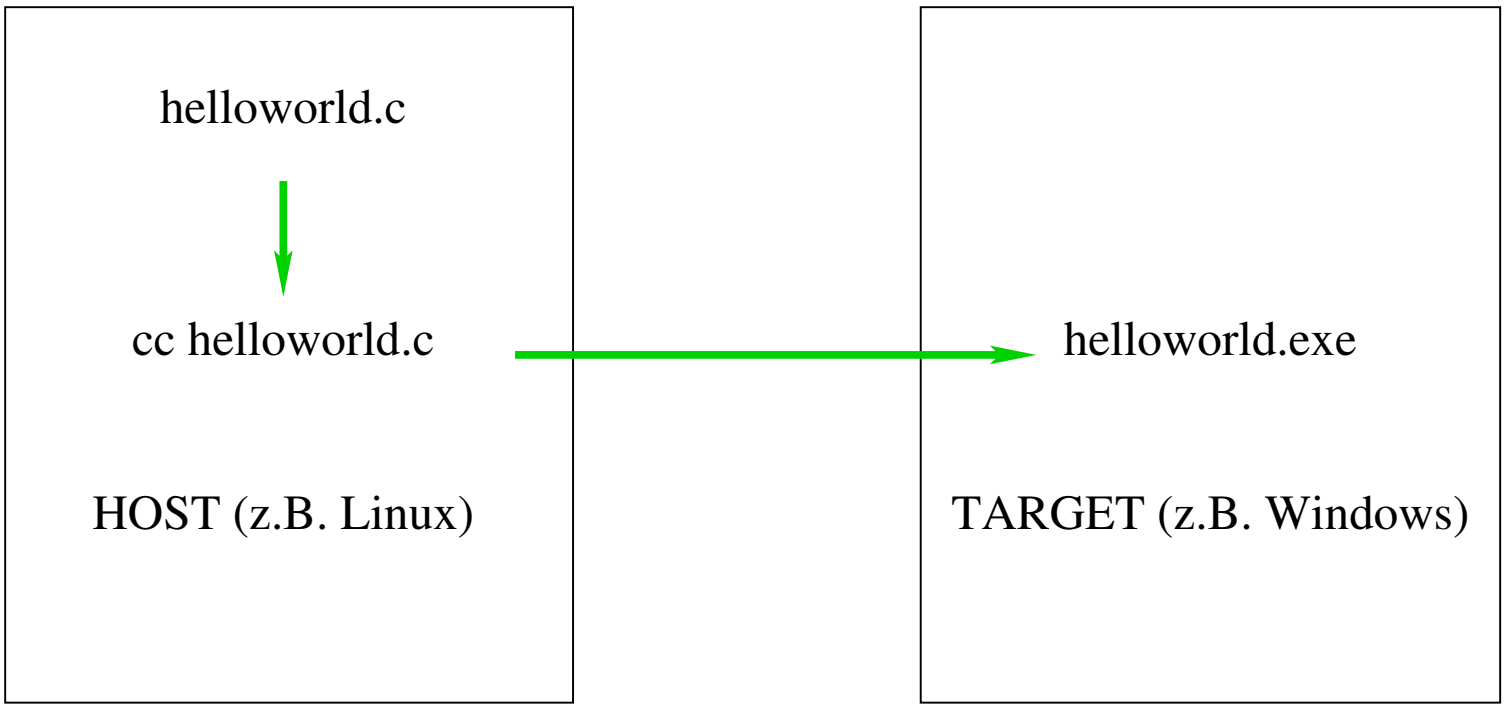
Wolfgang Dautermann

FH JOANNEUM

Chemnitzer Linuxtage 2011

Cross-Compiler

Ein Cross-Compiler ist ein Compiler, der auf einer Plattform läuft, aber Compilate (Executables) für eine andere Plattform erzeugt.



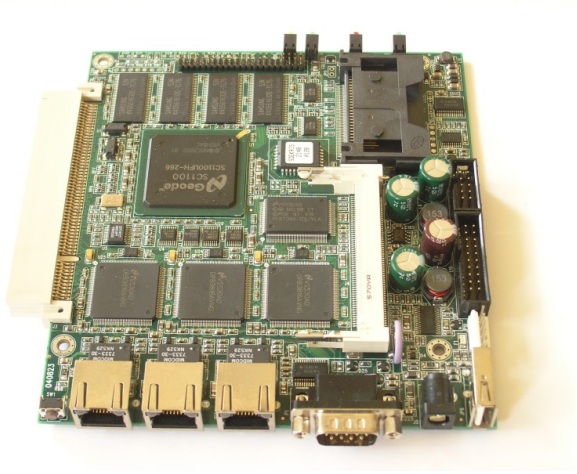
Cross-Compiler

Dabei können sich sowohl am HOST-Rechner als auch auf der TARGET-Plattform **alle** Komponenten unterscheiden:

- Betriebssystem
- CPU-Typ
- 32/64 Bit CPU (auch 16 oder 8 Bit)

Die Target-Plattform muss nicht mal real existieren!

Einsatzzwecke - Embedded Systeme



- Speicher/CPU-Ressourcen reichen für Compiler/Toolchain nicht aus
- Ein komfortable Entwicklungsumgebung läuft auf anderer Plattform[1]
- Zeitersparnis: anderer Rechner hat mehr Ressourcen und kann schneller compilieren.

[1] z.B. wird für embedded Systeme (oder auch für Smartphones) häufig eine Entwicklungsumgebung für PCs (z.B. Windows/Linux/MacOS) mitgeliefert

Einsatzzwecke

- Gewohnte Entwicklungsumgebung läuft nicht auf anderen Plattformen
- Build-Umgebung soll einheitlich sein
- Target-Plattform nicht verfügbar
- Softwarelizenz (Betriebssystem und/oder Entwicklungstools) nicht vorhanden
- (Noch) kein Compiler auf der Targetplattform verfügbar
- Kontinuierliche Integration – aktueller Softwarestand wird automatisch für diverse Plattformen compiliert, um Probleme früh zu erkennen.

Installation des Crosscompilers

...als Paket der Linuxdistribution

Debian

```
$ apt-get install mingw32
```

Opensuse[a]

[a] Leider nicht eindeutig, es gibt noch andere Repositories mit Crosscompilern, z.B. `http://download.opensuse.org/repositories/CrossToolchain:/mingw/`

`http://download.opensuse.org/repositories/windows:/mingw:/win32/openSUSE_11.3/`

als Repository dazugeben, dann (z.B.) das Paket `mingw32-cross-gcc` (C-Crosscompiler für Win32) installieren.

Crosscompilieren mit dem GCC

Die GNU Compiler Collection (GCC, `http://gcc.gnu.org/`) ist darauf ausgelegt, für möglichst viele Plattformen Compilate erzeugen zu können (lt. Wikipedia (`http://en.wikipedia.org/wiki/GNU_Compiler_Collection#Architectures`) ca. 43 CPUs). GCC eignet sich daher auch gut als Crosscompiler für viele Hosts und Target-Architekturen.

Compilieren des GCC für unterschiedliche Hosts/Targets

```
$ ./configure --host=HOST --target=TARGET
```

BUILDHOST $=$ HOST $=$ TARGET: native
BUILDHOST $=$ HOST $\neq$ TARGET: cross
BUILDHOST $\neq$ HOST $=$ TARGET: cross-native
BUILDHOST $\neq$ HOST $\neq$ TARGET: canadian-cross

Installation des Crosscompilers

...für Selbermacher (aufwendig!)

Man benötigt (Beispiel: für Windows):

- ein laufendes System mit einem (normalen) Compiler
- (aktuelle) Sourcen der GNU Binutils[2]
  `http://www.gnu.org/software/binutils/`
- (aktuelle) Sourcen des GCC `http://gcc.gnu.org/`,
  div. Hilfslibraries (libgmp, libmpfr, ...)
- MinGW Runtime + W32api
  `http://sourceforge.net/projects/mingw/`
- Viel Zeit![3]

[2] Assembler, Linker, div. Hilfsprogramme

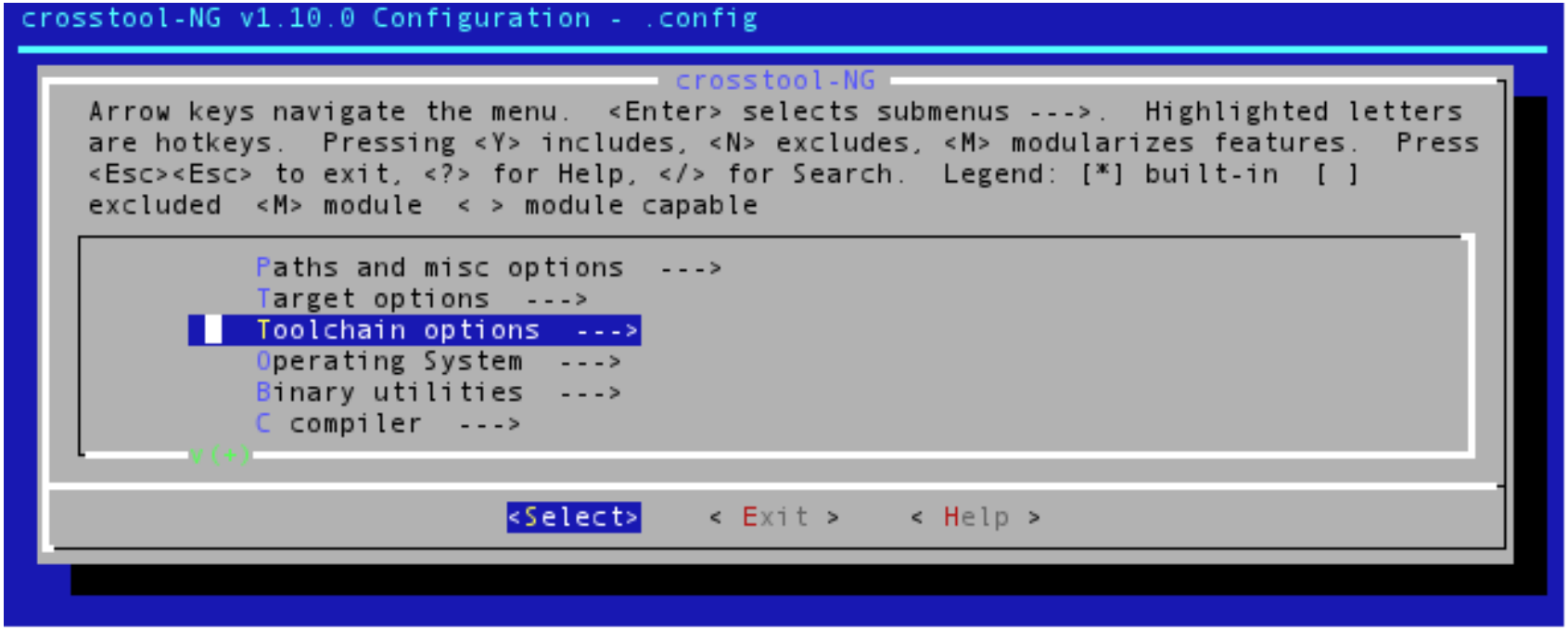
[3] zum Compilieren, Howtos lesen und Fehlermeldungen googlen ☺

Installation des Crosscompilers

crosstool-NG – toolchain generator

`http://ymorin.is-a-geek.org/projects/crosstool`

- `make menuconfig` (ähnlich Kernelkonfiguration)
- diverse supportete Targetarchitekturen (inkludierte Patches!)



Hello world unter Linux compilieren

Compilieren von helloworld.c

```
/* Helloworld Programm */
#include <stdio.h>
int main()
{
        printf("Hello world!\n");
        return(0);
}

$ gcc -Wall -o helloworld helloworld.c
$ file helloworld
helloworld: ELF 32-bit LSB executable [...] for GNU/Linux [...]
```

erzeugt ein normales Linux-Executable.

Hello world unter Linux compilieren (32/64-Bit)

Biarch Compiler

Auf 64-Bit Linux-Systemen ist gcc defaultmässig so installiert, dass er sowohl 32 als auch 64 Bit Executables erzeugen kann:

Compilieren von helloworld.c

```
$ gcc -Wall -o helloworld helloworld.c
$ file helloworld
helloworld: ELF 64-bit LSB executable, x86-64, [...]
$ gcc -m32 -Wall -o helloworld helloworld.c
$ file helloworld
helloworld: ELF 32-bit LSB executable, Intel 80386, [...]
```

Die Option `-m32` gibt an, dass ein 32 Bit Binary erzeugt werden soll (defaultmäßig wird ein 64-Bit Binary erzeugt).

Hello world unter Linux compilieren (32/64-Bit)

...auf einer 32-Bit Maschine

Debian/Ubuntu: `gcc-multilib` installieren, dann geht das auch umgekehrt:

Compilieren von helloworld.c

```
$ gcc -Wall -o helloworld helloworld.c
$ file helloworld
helloworld: ELF 32-bit LSB executable, Intel 80386, [...]
$ gcc -m64 -Wall -o helloworld helloworld.c
$ file helloworld
helloworld: ELF 64-bit LSB executable, x86-64, version 1 [...]
```

Die Option `-m64` gibt an, dass ein 64 Bit Binary erzeugt werden soll (defaultmäßig wird ein 32-Bit Binary erzeugt).

Hello world unter Linux **für Windows** compilieren

Cross-Compiler ist installiert unter `/usr/bin/i686-pc-mingw32-gcc`.
Einfach den speziellen Compiler auswählen und damit compilieren:

Compilieren von helloworld.c

```
$ i686-pc-mingw32-gcc -Wall -o helloworld.exe helloworld.c
$ file helloworld.exe
helloworld.exe: MS-DOS executable PE  for
                MS Windows (console) Intel 80386 32-bit
```

erzeugt ein Windows-Executable.

Umfangreichere Programme unter Linux **für Windows** compilieren

Die notwendigen Bibliotheken (Libraries) müssen auch für die Zielplattform verfügbar sein[4]. Ansonsten kann ganz normal compiliert werden.

Compilieren von helloworld-sdl.cpp

```
$ i686-pc-mingw32-gcc -Wall -o helloworld-sdl.exe helloworld-sdl.cpp
      $(/usr/i686-pc-mingw32/sys-root/mingw/bin/sdl-config --libs --cxxflags)
$ file helloworld-sdl.exe
helloworld-sdl.exe: PE32 executable for MS Windows (GUI)
Intel 80386 32-bit
```

erzeugt ein Windows-Executable.

[4] entweder selbst übersetzen oder fertig dowloaden

Beispiel - Buildprozess mit Cmake

Unverändertes(!) `CMakeLists.txt`

Toolchain-mingw32.cmake

```
# the name of the target operating system
SET(CMAKE_SYSTEM_NAME Windows)

# which compilers to use for C and C++
SET(CMAKE_C_COMPILER i686-pc-mingw32-gcc)
SET(CMAKE_CXX_COMPILER i686-pc-mingw32-g++)

# here is the target environment located
SET(CMAKE_FIND_ROOT_PATH /usr/i686-pc-mingw32/sys-root/i686-pc-mingw32)

# adjust the default behaviour of the FIND_XXX() commands:
# search headers and libraries in the target environment,
# search programs in the host environment
set(CMAKE_FIND_ROOT_PATH_MODE_PROGRAM NEVER)
set(CMAKE_FIND_ROOT_PATH_MODE_LIBRARY ONLY)
set(CMAKE_FIND_ROOT_PATH_MODE_INCLUDE ONLY)
```

```
cmake -DCMAKE_TOOLCHAIN_FILE=~/Toolchain-mingw32.cmake SRCPATH
```

Installationspakete für Windows

Das Nullsoft Scriptable Install System (`http://nsis.sourceforge.net/`) ist freie Software. Es läuft nicht nur unter Windows sondern kann auch (nativ) unter Linux/Unix installiert werden:

NSIS Installation

```
$ apt-get install nsis # Debian/Ubuntu
$ zypper install mingw32-cross-nsis # Opensuse
```

(oder selbst compilieren)

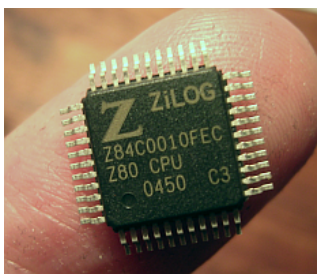
Damit können auch Windows `setup.exe`'s erzeugt werden (`makensis helloworld.nsi`).

SDCC - Small Device C Compiler

http://sdcc.sourceforge.net/

Compiler für Linux/Windows/MAC (und andere Betriebssysteme) unterstützt die folgenden CPUs:

- Intel 8051
- Maxim 80DS390
- Zilog Z80
- Motorola 68HC08



cc65 - the 6502C compiler

http://www.cc65.org/

Compiler für Linux/Windows/MAC (und andere Betriebssysteme) für die MOS Technology 6502 CPU – und damit die folgenden Systeme

- Commodore C64, C128, VC16, Plus4
- Commodore P500, CBM600, CBM700
- Apple II
- Nintendo Entertainment System (NES)
- ...

Fragen? Feedback?

Kurze Live-Demonstration (wenn gewünscht)...

Vielen Dank für Ihre Aufmerksamkeit

Wolfgang Dautermann
wolfgang.dautermann@fh-joanneum.at

Credits für Bilder

- http://de.wikipedia.org/w/index.php?title=Datei:Z84C0010FEC_LQFP.png (Public Domain)
- http://commons.wikimedia.org/wiki/File:Soekris_net4801_board.jpg (Fotograf: Jaho)
- http://de.wikipedia.org/w/index.php?title=Datei:MOS_6502AD_4585_top.jpg (Fotograf: Dirk Oppelt)
- http://nsis.sourceforge.net (NSIS Logo)