

# Praxis Cross-Plattform Entwicklung

Echte Linux Applikationen unter  
Windows erstellen.

*Friedhelm Wolf, Homag AG*  
1. April 2005

# Homag AG



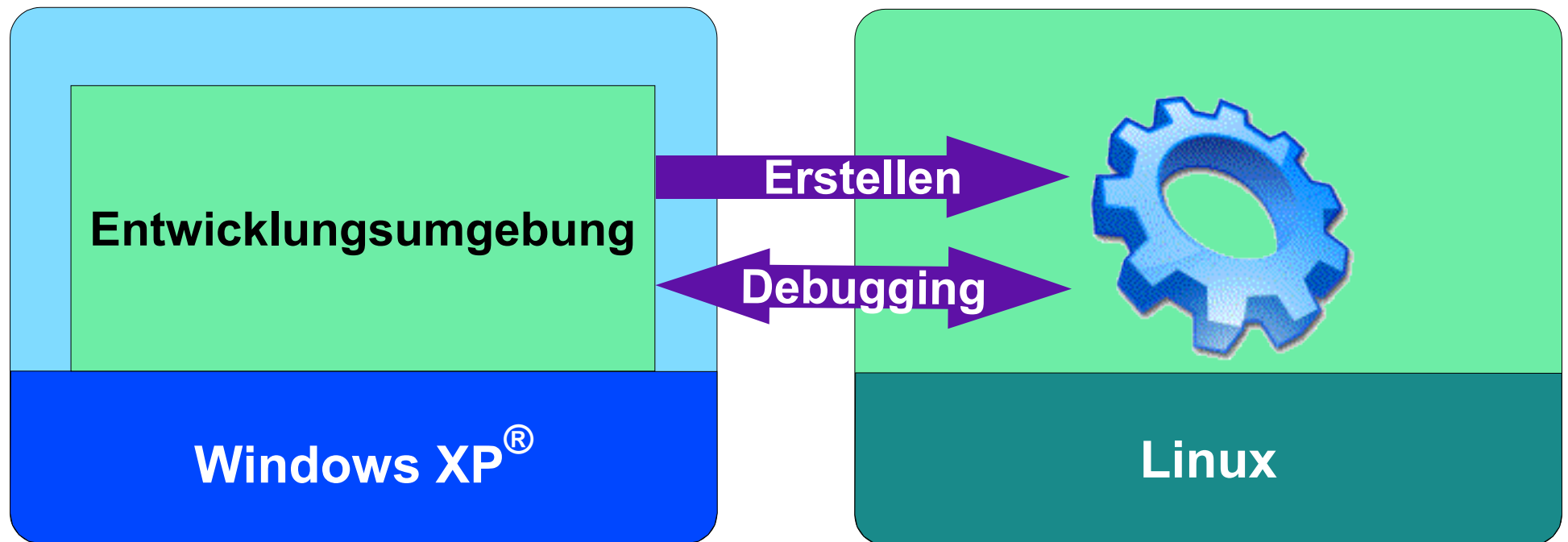
# Überblick

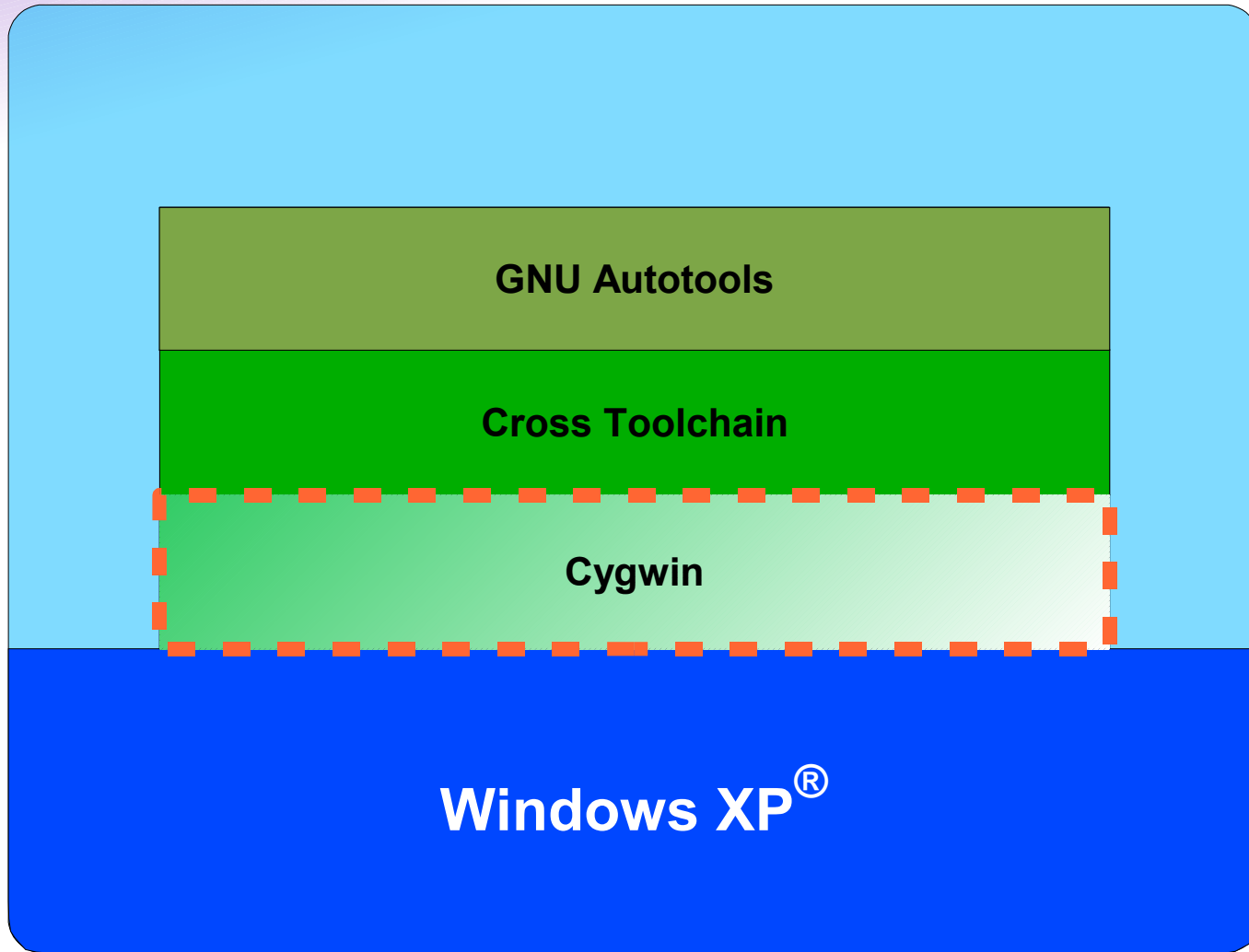
1. Motivation
2. Gesamtsystem
3. Cygwin
4. Cross-Toolchain
5. GNU Autotools
6. Ausblick

# Motivation

- Evaluierung von Linux im Echtzeitbereich
- Entwickler sind Windows<sup>®</sup> ( Visual Studio ) gewohnt
  - Linux im bekannten Umfeld nutzen
- Erhaltung vorhandener Entwicklungsprozesse
- Ziel: Entwicklung von Linux Applikationen in einer Windows<sup>®</sup> Entwicklungsumgebung

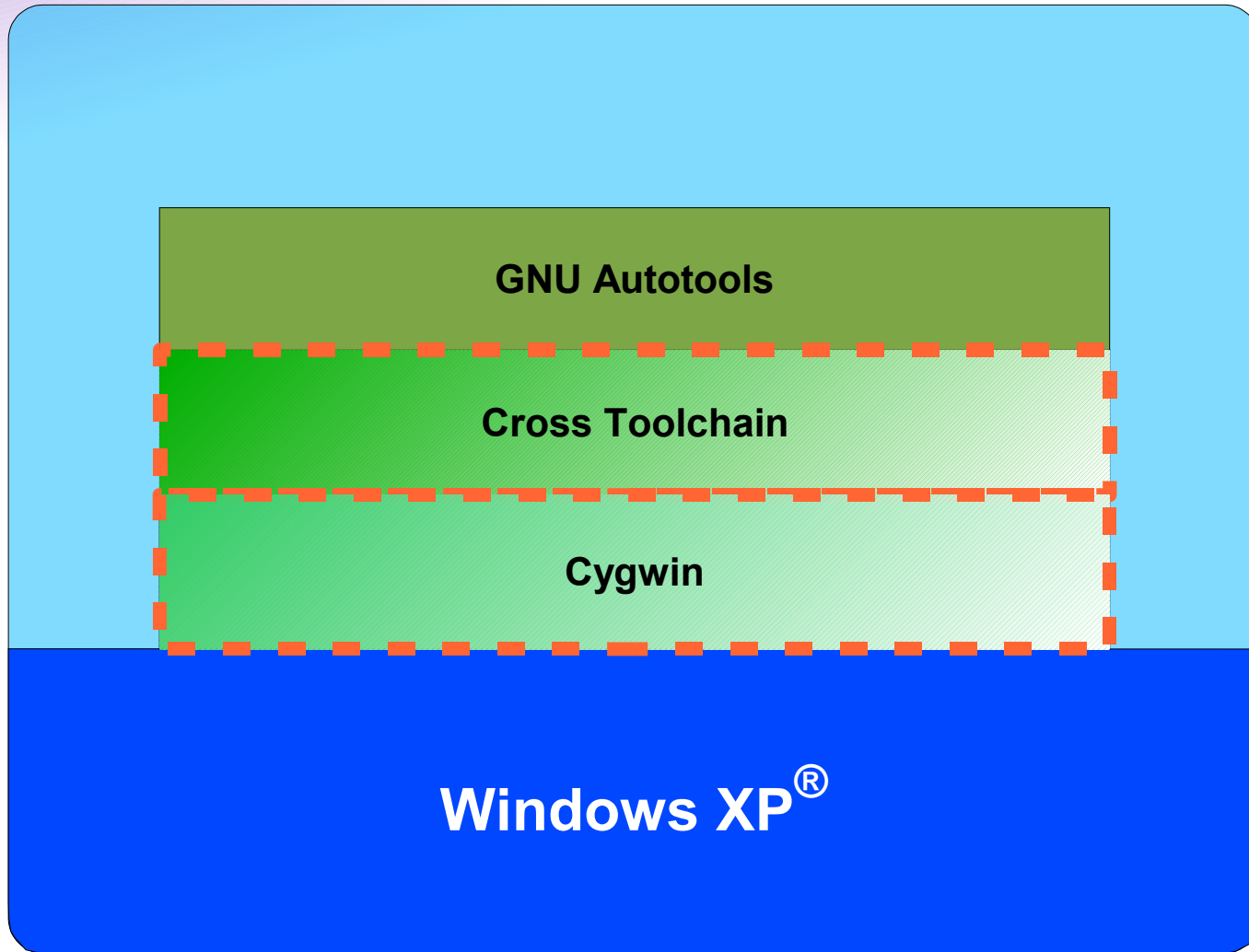
# Gesamtsystem





# Cygwin

- Abbildung der UNIX<sup>®</sup> API als Windows<sup>®</sup> DLL
- Sammlung der wichtigsten GNU Pakete und eigene Paket-Verwaltung
- Installation über das Internet möglich
- Entwicklung von Executables (“.exe” Format) mit MinGW und GCC
- Homepage : <http://www.cygwin.com/>



# Definition

“Cross compiling is the procedure for building a program for a platform different from the one on which the cross compiler runs.

'Platform' does not only mean the hardware architecture but also software platforms.”

Quelle: <http://www-public.tu-bs.de:8080/~y0018536/dc/online/node5.html>

# Cross Toolchain

## - Prinzip -

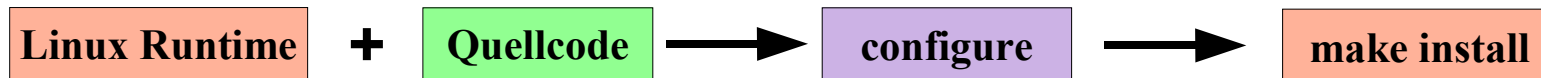
- Erstellen der GNU Compiler Collection (GCC) aus dem Quellcode
- Spezielle Konfiguration für eine bestimmte Zielplattformen
- Zielplattformen unterscheiden sich in Prozessorarchitektur und Betriebssystem
- Man unterscheidet Host, Build und Target

# Cross Toolchain

## - Verzeichnisstruktur -



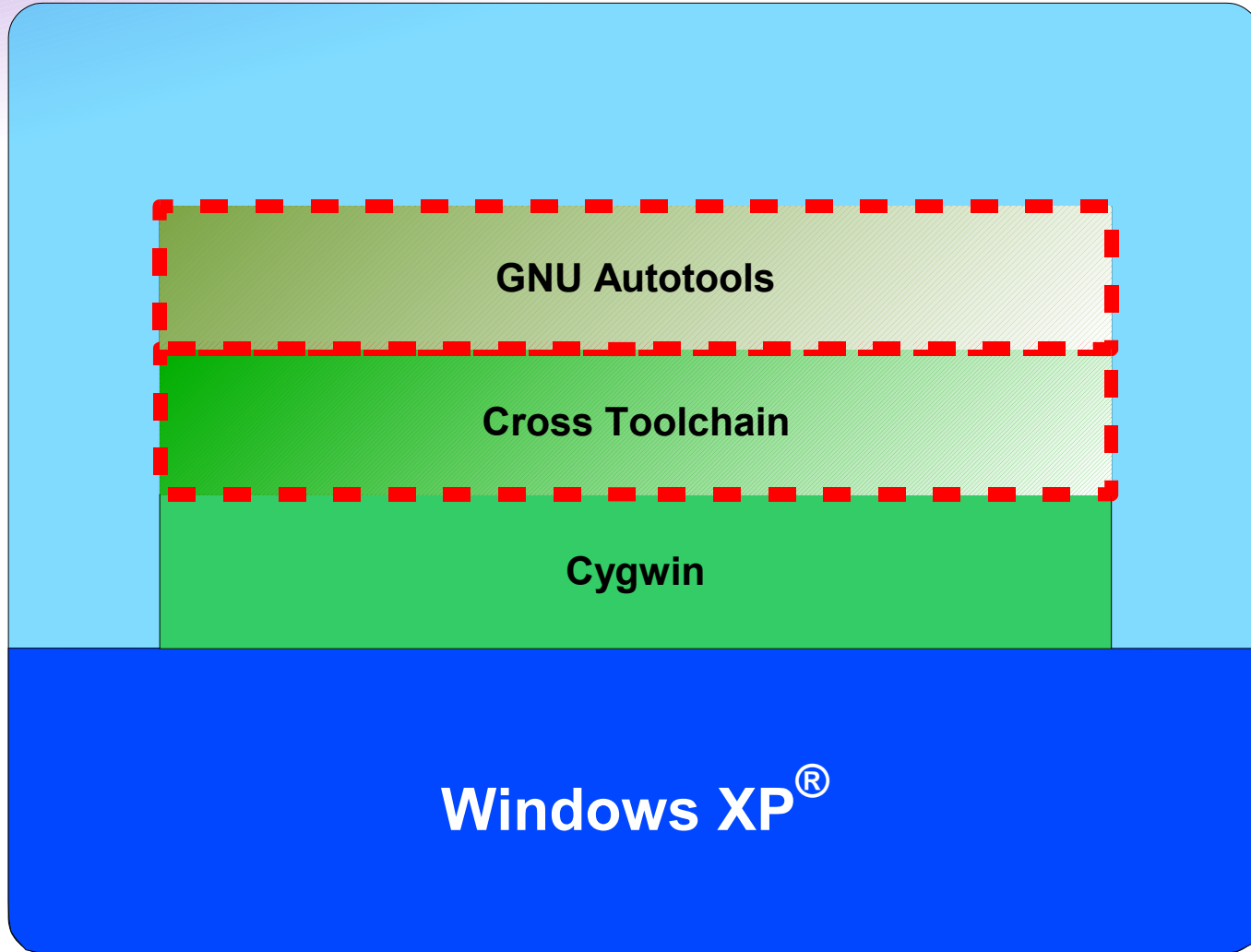
## i686-pc-cygwin



# Cross Toolchain

## - Vorgehen -

1. System Header und Bibliotheken auf das Build System holen
2. Quellcode der benötigten Pakete downloaden
3. Binutils erstellen und installieren
4. GCC erstellen und installieren
5. GDB erstellen und installieren

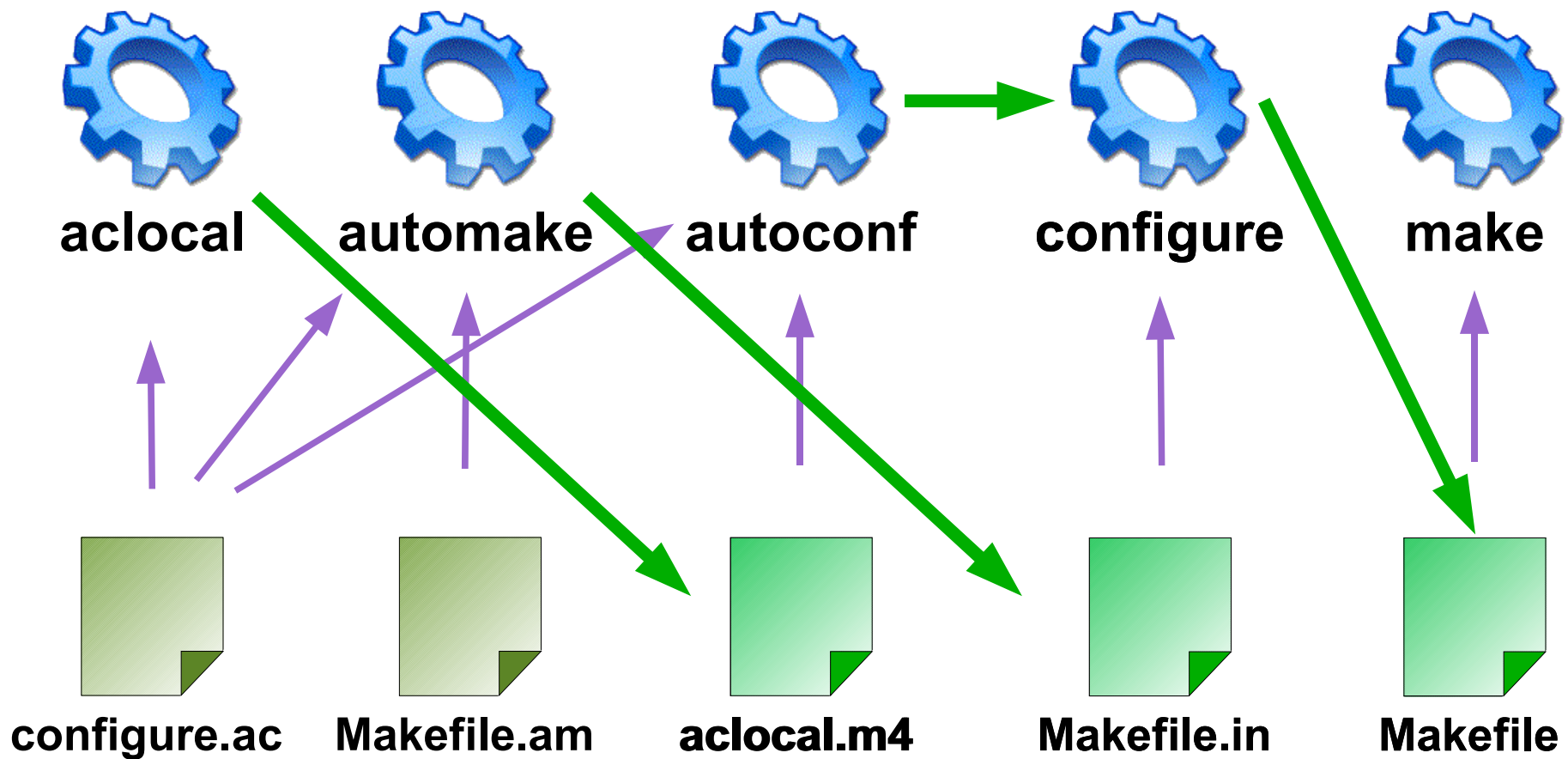


# GNU Autotools

- Besteht aus verschiedenen Programmen des GNU Projektes, die eng zusammenarbeiten
- Beschreibung einer Projekt-Konfiguration auf plattformunabhängiger Basis
- Erzeugung eines angepassten GNU Makefiles
- Verwaltung des Build-Vorgangs von der Konfiguration über Kompilierung und Linken bis zur Installation

# GNU Autotools

- Ablauf -



# GNU Autotools

## - Programme -

- **Autoconf** – Makrosprache zur Erstellung eines “configure”-Scripts
- **Automake** – Erstellen von Makefiles
- **Libtool** – Abstraktion für Bibliotheksverwaltung
- **Configure** – Prüfung der Systemgegebenheiten
- **Make** – Steuerung der Erstellung von Executables und anderer Dateien aus Quellcodedateien.

# GNU Autotools

## - Makefile.am -

```
bin_PROGRAMS = program1
lib_LTLIBRARIES = lib1

program1_CFLAGS = -DMACRO_2 ...
program1_LDADD = -lpthread ...
program1_SOURCES = file1.cpp ...

lib1_SOURCES = libfile1.cpp ...
...

clean : rm -rf *.o
```

# GNU Autotools

## - configure.ac -

```
AC_INIT(program1, 1.0)
```

```
AC_CANONICAL_SYSTEM
```

```
AM_INIT_AUTOMAKE
```

```
AC_PROG_CC(i686-pc-linux-gnu-gcc)
```

```
AC_PROG_CXX(i686-pc-linux-gnu-g++)
```

```
AC_PROG_LIBTOOL
```

```
AC_OUTPUT(Makefile)
```

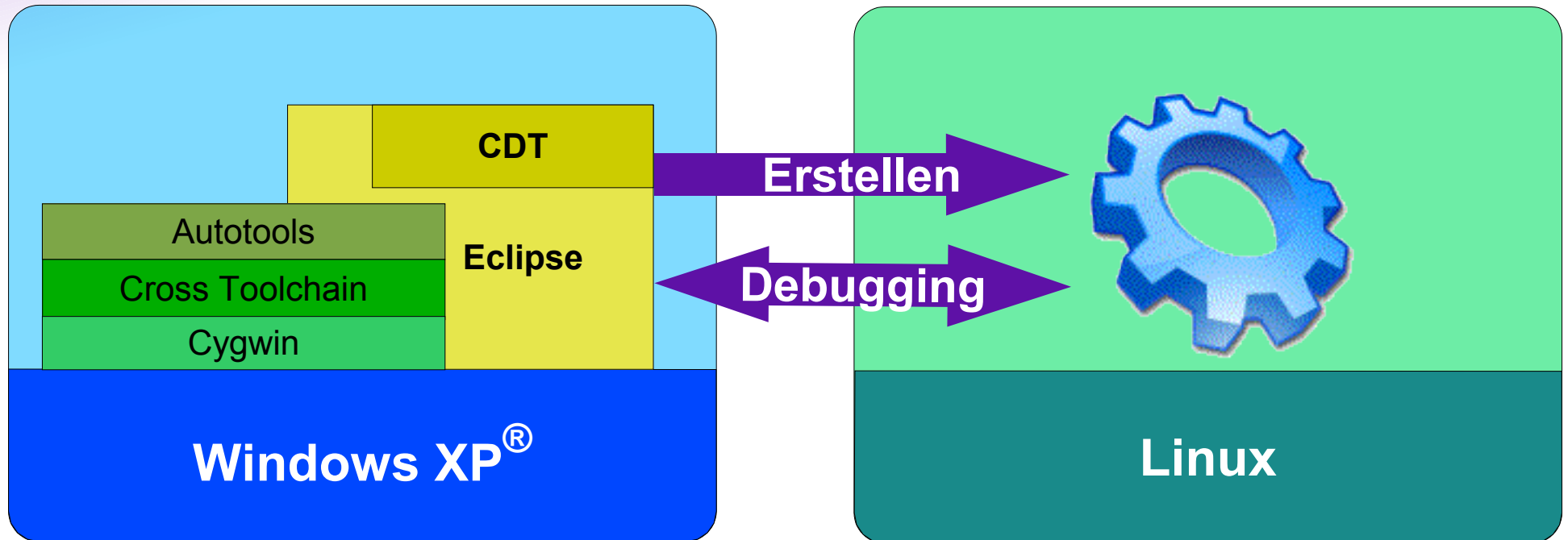
# GNU Autotools

## - Bootstrap Script -

```
aclocal
libtoolize --force
automake --add-missing --foreign
autoconf

mkdir -p debug
cd debug
../configure CFLAGS="-g -O0 -Wall" \
             CXXFLAGS="-g -O0 -Wall" \
             --build=i686-pc-cygwin \
             --host=i686-pc-linux-gnu
make
```

# Ausblick



- Vortrag “C / C++ Crossdevelopment mit Eclipse” von Jan Altenberg

# Quellen

- Vaughan, Elliston, Tromeu und Taylor:  
**GNU Autoconf, Automake, and Libtool.**  
New Riders Publishing, 2001
- <http://billgatliff.com/~bgat/twiki/bin/view/Crossgcc/WebHome>
- <http://www.kegel.com/crosstool>
- <http://www.gnu.org> ( Manuals der einzelnen GNU Tools )
- <http://vipe.technion.ac.il/~shlomif/lecture/Autotools/>  
( Autotools Tutorial )

# Kontakt

Friedhelm Wolf  
(Abteilung Software-Pool)

Homagstraße 3-5  
D-72296 Schopfloch

Telefon: 07443 / 13-3126  
Email: [friedhelm.wolf@homag.de](mailto:friedhelm.wolf@homag.de)