

Developing for Embedded Linux on Windows

Dr. Peter Schojer (peter.schojer@appinf.com)

Applied Informatics
Software Engineering GmbH

Overview

- > Motivation
- > Required Software
- > Creating a Cross Compiler
- > Eclipse
- > Debugging With Eclipse
- > Summary

Motivation – why Eclipse?

- > free
- > open-source
 - > write your own plugins
 - > many supporters
- > Windows, Linux, Mac OS X
- > Java, C, C++, COBOL, PHP
- > embedded to enterprise applications

Motivation – why Windows?

- > if you can use Linux, use it
 - > VMware, ...
 - > work remotely
- > IT constraints
 - > no VMware license
 - > single OS policy: Windows only
- > most developers are used to Windows

Motivation – why embedded

- > growing market
- > devices get more powerful
 - > software gets more complex
- > Conclusion
 - > more developers needed
 - > so even if you don't want to develop for embedded,...

Required Software

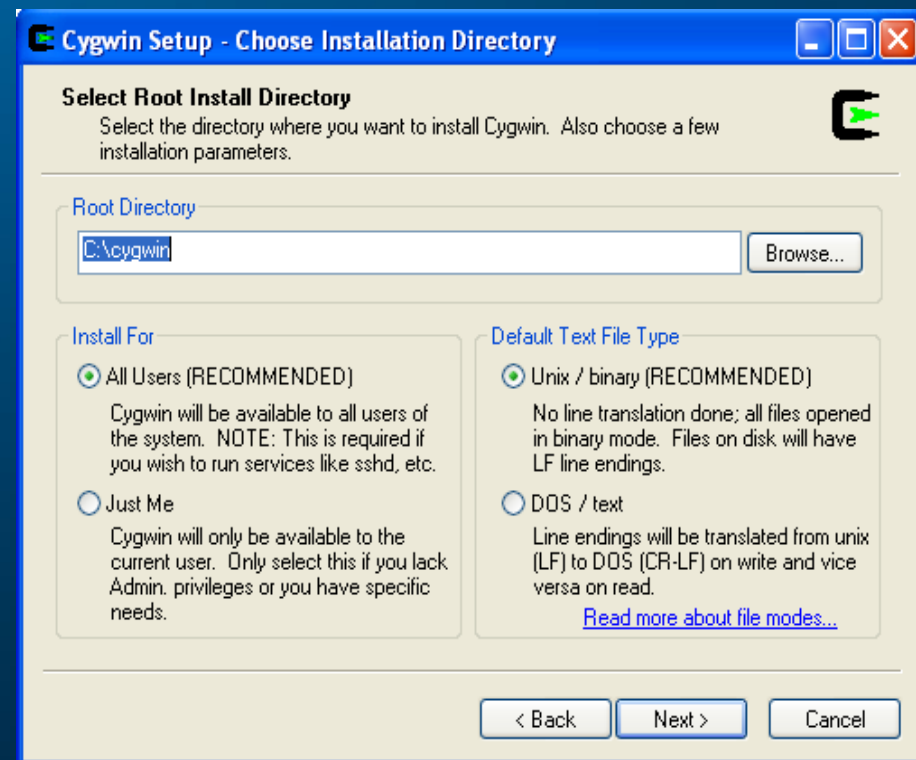
- > Cygwin
- > Eclipse with CDT
- > Cross Toolchain
 - > Compiler
 - > Linker
 - > Binutils
 - > Debugger

Cygwin

- > (imperfect) UNIX emulation layer for Windows
 - > no locale support
 - > file system restrictions
- > recompiling UNIX applications enough to make them run on Windows
 - > like gcc

Cygwin – Install

- > www.cygwin.com
- > additional packages
 - > Web->{wget}
 - > Utils->{diffutils, patch}
 - > some of the Devel packages



Cygwin – Restriction

- > Cygwin is only an application running on Windows
 - > file system restriction
 - > upper/lower case
 - > access rights, executable flag (.sh, .exe)
 - > line endings
 - > text mode: cygwin apps (e.g. svn) convert
 - > binary mode: no conversion

Cygwin – Line Endings

- > Text or Binary?
 - > <http://cygwin.com/cygwin-ug-net/using-textbinary.html>
 - > recommended to use binary
 - > textmode: on write `\n` -> `\r\n`
 - > fseek?
 - > binary
 - > Makefile/scripts MUST not contain `\r\n`

Cygwin – File System Restr.

- > bypass them by using *managed* directories
 - > mount -o managed
c:\cygwin\safe /home/\$(USER)/safe

Cross Compiler – Tools

> Crosstool

> <http://code.google.com/p/crosstool/>

> Ptxdist

> http://www.pengutronix.de/software/ptxdist/index_en.html

> Crosstool NG

> <http://ymorin.is-a-geek.org/dokuwiki/projects/crosstool>

Crosstool NG

- > very recent configurations
 - > GCC 4.2.1, glibc 2.6.1
 - > only the latest are supported
 - > no gcc 3.x, glibc < 2.5
- > arm, x86, mips
- > Cygwin support very rudimentary

Ptxdist

- > doesn't build out-of-the box in Cygwin
 - > but it builds :-)
- > arm, x86, mips, powerpc
- > recent configurations
 - > glibc 2.3.6, 2.5, gcc 4.x
- > works good for ARM targets on Cygwin
 - > others not tested

Crosstool

> by Dan Keigel
(<http://keigel.com/>)



Crosstool

- > lots of configurations
- > only few recent configurations
- > NPTL support just added
 - > must use SVN version
- > but good Cygwin support!
 - > no guarantees though

Crosstool – Usage

- > decompress/checkout to a *managed* Cygwin directory
- > choose your demo-<arch>.sh file and select a configuration
- > execute the shell script
- > wait a few hours!
 - > result will be in /opt/crosstool

Crosstool – Example

`BINUTILS_DIR=binutils-2.16.1`

`GCC_CORE_DIR=gcc-4.2.0`

`GCC_DIR=gcc-4.2.0`

`GLIBC_DIR=glibc-2.5`

`LINUX_DIR=linux-2.6.15.4`

`LINUX_SANITIZED_HEADER_DIR=linux-libc-headers-2.6.12.0`

`GDB_DIR=gdb-6.6`

Crosstool – Example (2)

```
GLIBC_EXTRA_CONFIG="$GLIBC_EXTRA_CONFIG --  
with-tls --with-__thread --enable-  
kernel=2.4.18"
```

```
GLIBC_ADDON_OPTIONS="=nptl"
```

demo-arm.sh:

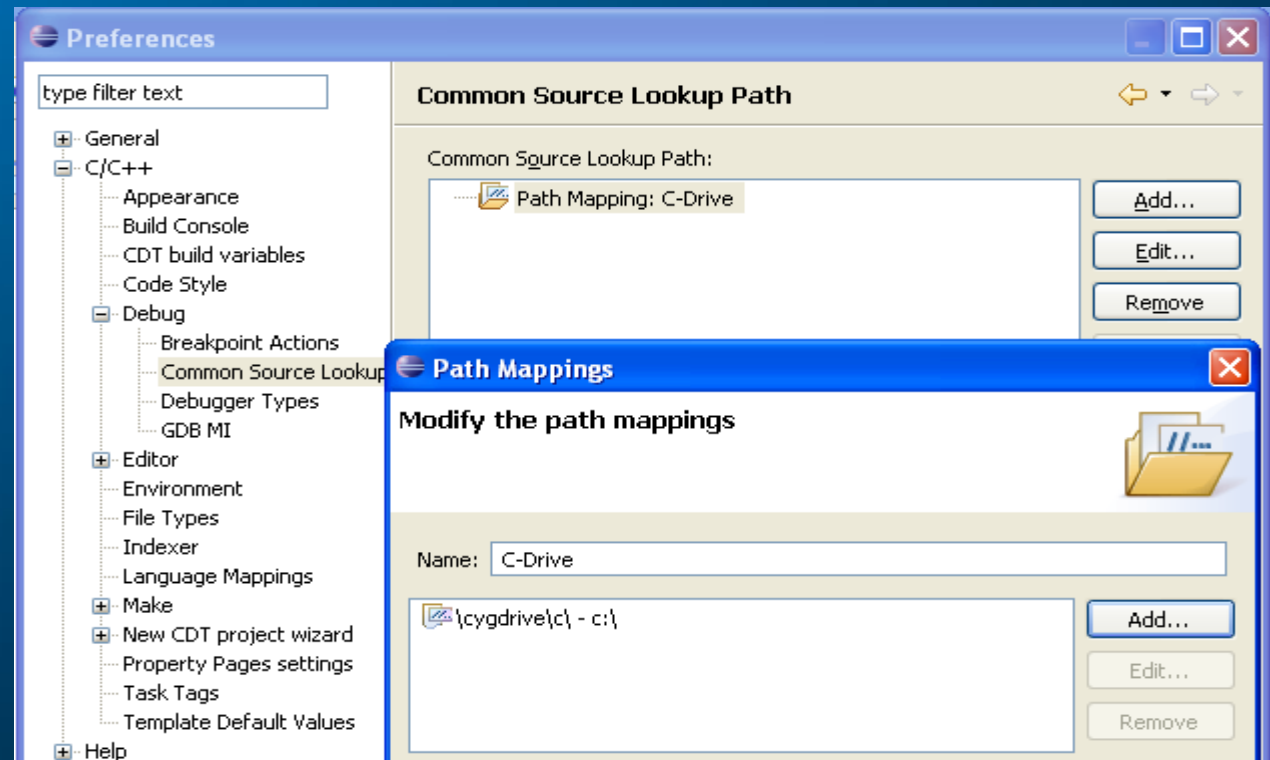
```
eval `cat arm.dat gcc-4.2.0-glibc-2.5-nptl.dat` sh  
all.sh --notest --gdb
```

Eclipse

- > IDE for various programming languages
 - > we only use C/C++
- > supports remote debugging
- > uses gcc as compiler
- > <http://www.eclipse.org>
 - > Eclipse for C/C++ developers
 - > current version: 3.3

Eclipse – Setup

> Path Settings



Eclipse – Additional Plugins

- > Device Software Development Platform
 - > <http://www.eclipse.org/dsdp/>
 - > remote file system support
 - > improved remote debugging
- > Subversion plugin
 - > <http://subclipse.tigris.org/>

Eclipse – Projects

- > two types of projects
 - > managed projects
 - > Makefile projects
 - > simply reuse your existing Linux build system on Windows
 - > faster and easier for varying cross compile targets than creating Eclipse configurations
 - > use Cygwin paths! (e.g.: /cygdrive/c/home)

Sample Makefile

```
TOOL= /cygdrive/c/cygwin/opt/crosstool/gcc-4.1.0-glibc-2.3.2/arm-unknown-linux-gnu/bin/arm-unknown-linux-gnu
```

```
CC      = $(TOOL)-gcc
```

```
CXX     = $(TOOL)-g++
```

```
LINK    = $(CXX)
```

```
STRIP   = $(TOOL)-strip
```

```
...
```




Project Explorer

- Hello
 - Binaries
 - Hello.exe - [arm/le]
 - Includes
 - Display.h
 - lcd_lib.h
 - Source
 - Display.cpp
 - Hello.cpp
 - lcd_lib.c
 - Output
 - Display.o - [arm/le]
 - Hello.o - [arm/le]
 - lcd_lib.o - [arm/le]
 - Makefile

```

//-----
// Name      : Hello.cpp
// Author    : Peter Schojer
// Version   :
// Copyright : Your copyright notice
// Description: Hello World in C, Ansi-style
//-----

#include "Display.h"

int main(void) {
    Display& dp = Display::instance();
    dp.clearDisplay();
    dp.writeLine("Write some extra long hello world");
    return EXIT_SUCCESS;
}

```

C/C++

Outlin

- Display.h
 - main(void) : int

Problems Tasks Console Properties

C-Build [Hello]

```

**** Build of configuration Default for project Hello ****

make all
/cygdrive/c/cygwin/opt/crosstool/gcc-4.1.0-glibc-2.3.2/arm-unknown-linux-gnu/bin/arm-unknown-linux-gnu-g++ -g -D_DEBUG
-static -Wall -fmessage-length=0 -c -o Display.o Display.cpp
/cygdrive/c/cygwin/opt/crosstool/gcc-4.1.0-glibc-2.3.2/arm-unknown-linux-gnu/bin/arm-unknown-linux-gnu-gcc -c -o
lcd_lib.o lcd_lib.c
/cygdrive/c/cygwin/opt/crosstool/gcc-4.1.0-glibc-2.3.2/arm-unknown-linux-gnu/bin/arm-unknown-linux-gnu-g++ -o Hello.exe
Hello.o Display.o lcd_lib.o

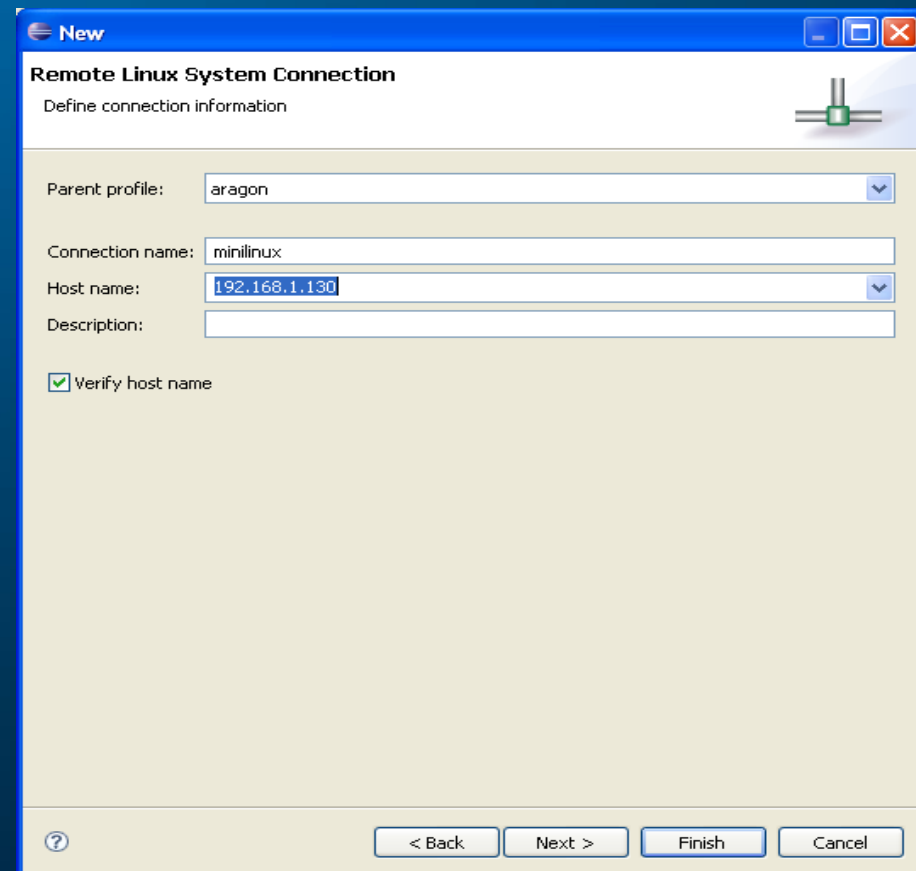
```

Eclipse Debugging

- > Device Software Development
 - > one big disadvantage
 - > dstore server needed on target
 - > requires JRE on the target, Perl recommended
- > No DSDP
 - > remote debugging still possible
 - > more manual steps required

Debugging with DStore

- > Create a Connection to the Target
- > Create a remote debug configuration and set the cross gdb (next page)





Create, manage, and run configurations



type filter text

- C/C++ Attach to Local Application
- C/C++ Local Application
- C/C++ Postmortem debugger
- C/C++ Remote Application
 - Hello

Filter matched 5 of 5 items

Name: Hello

- Main
- Arguments
- Debugger
- Source
- Common

Connection: emblinux New

Project: Hello Browse...

C/C++ Application: Hello Search Project... Browse...

Remote Path for C/C++ Application: /root

Skip download to target path.

Apply Revert

Debug Close



Debugging Without DStore

- > create a connection to the target
- > upload the binary via the connection
 - > check file is executable
- > telnet/ssh to target
 - > *gdbserver winhost:gdbserverport executable [args]*
- > create a debug configuration as shown on the next slide

Create, manage, and run configurations



type filter text

- C/C++ Attach to Local Ap
- C/C++ Local Application
 - Hello
- C/C++ Postmortem debug

Filter matched 4 of 4 items

Name: Hello

Main Arguments Environment **Debugger** Source Common

Debugger: gdbserver Debugger

Stop on startup at: main Advanced...

Debugger Options

Main Shared Libraries Connection

GDB debugger: C:\cygwin\opt\cross\gcc-4.1.0-glibc-2.3.2\arm-unkno Browse...

GDB command file: .gdbinit Browse...

(Warning: Some commands in this file may interfere with the startup operation of the debugger, for example "run".)

GDB command set: Standard

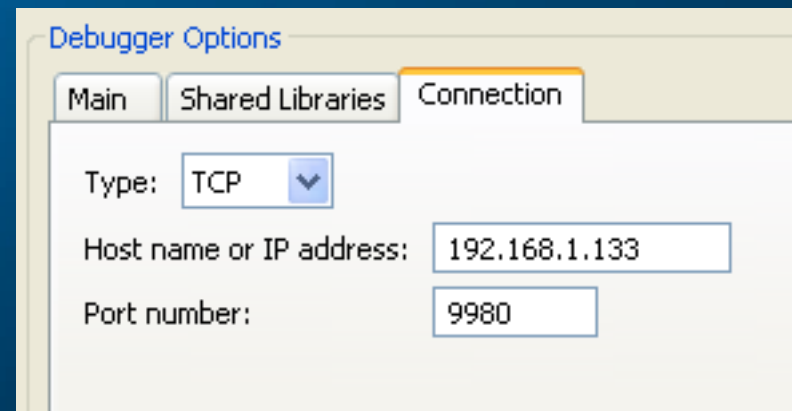
Protocol: mi

Verbose console mode

Apply Revert

Debugging Without DStore

- > update the debug connection settings to point to the target
- > start debugging





Debug Console

```

Hello_s [C/C++ Local Application]
├── gdbserver Debugger (03.09.07 13:16) (Suspended)
│   └── Thread [0] (Suspended)
│       └── 1 main() \cygdrive\c\projects\Hello\Hello.cpp:13 0x00008274
├── C:\cygwin\opt\cross\gcc-4.1.0-glibc-2.3.2\arm-unknown-linux-gnu\bin\arm-unknown-linux-gnu-gdb.exe (03.09.07 13:16)
└── C:\projects\Hello\Hello (03.09.07 13:16)
    
```

Variables

Name	Value
dp	0x2f727375
lines	
std::_List_base<std::basic_string<char, std::ch	
_M_impl	{...}
std::allocator<std::_List_node<std::bas	{...}
_M_node	

```

//-----
// Name      : Hello.cpp
// Author    : Peter Schojer
// Version   :
// Copyright : Your copyright notice
// Description : Hello World
//-----

#include "Display.h"

int main(void) {
    Display& dp = Display::instance();
    dp.clearDisplay();
    dp.writeLine("Write some extra long hello world");
    return EXIT_SUCCESS;
}
    
```

Outline

- Display.h
 - main(void) : int

Summary

- > we provided a short introduction to remote debugging with Eclipse
 - > pointers to required software
- > remote debugging with Eclipse is not yet perfect
 - > JRE on embedded often not available
 - > without it debugging is cumbersome
 - > better (commercial) alternatives exist