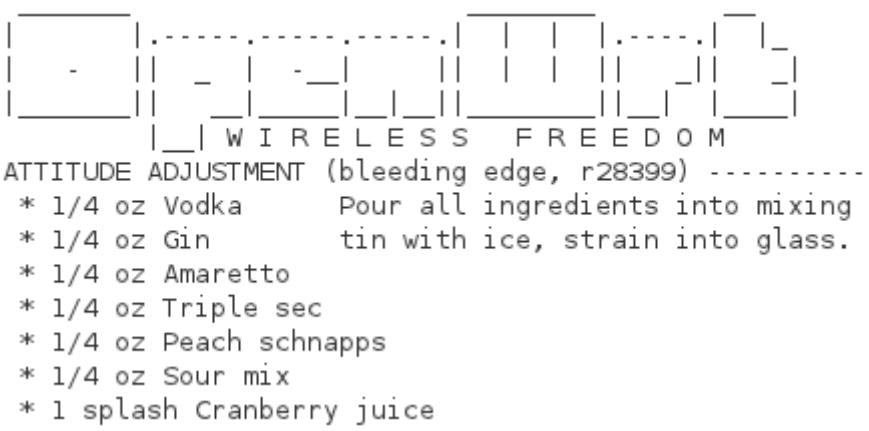
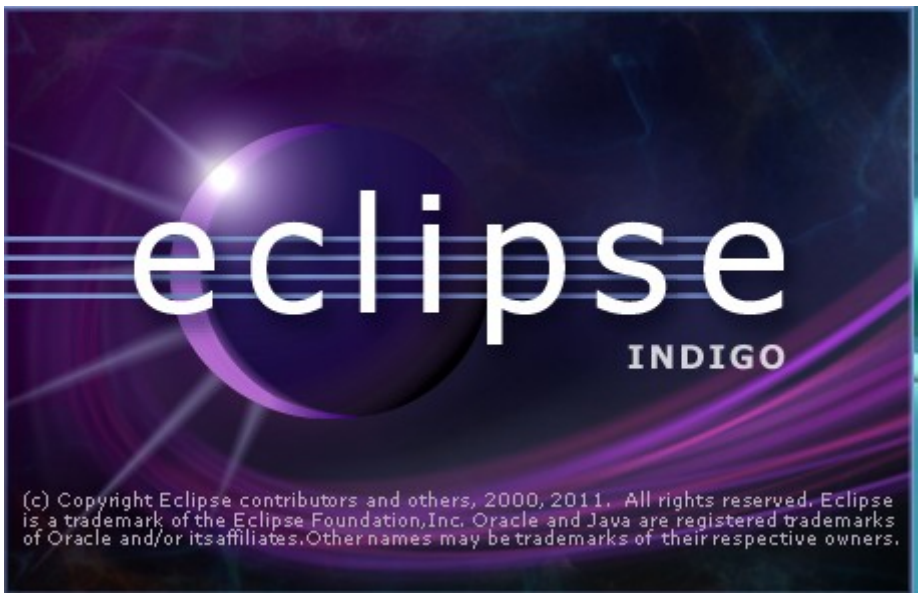
OpenWrt C/C++ Development

with Eclipse





by Jens Köhler

Table of Contents

1 Copyright

2 Purpose of document

With Eclipse your are able to develop software for OpenWrt target devices in a very comfortable manner. Eclipse provides a complete development suite.

This document explains how to use Eclipse C/C++ IDE 3.7 (Indigo) with OpenWrt's cross toolchain, how to setup remote target device source level debugging and remote access via eclipse. It is shown how to write, compile and debug programs for OpenWrt target devices.

3 OpenWrt Prerequisites

Install OpenWrt Buildroot then type

```
tieto@vubuntu:~/openwrt/trunk$ make menuconfig
```

and check

- enable (*) Build the OpenWrt SDK
- enable (*) “Build the OpenWrt based Toolchain”
- enable (*) “Advanced configuration options (for developers)->Toolchain Options
  → ”gdb”
  → “Build/install c++ compiler and libstdc++” (if C++ is required)

If not already done execute

```
make toolchain/install
```

4 Target Prerequisites

The following packages are required on your target device:

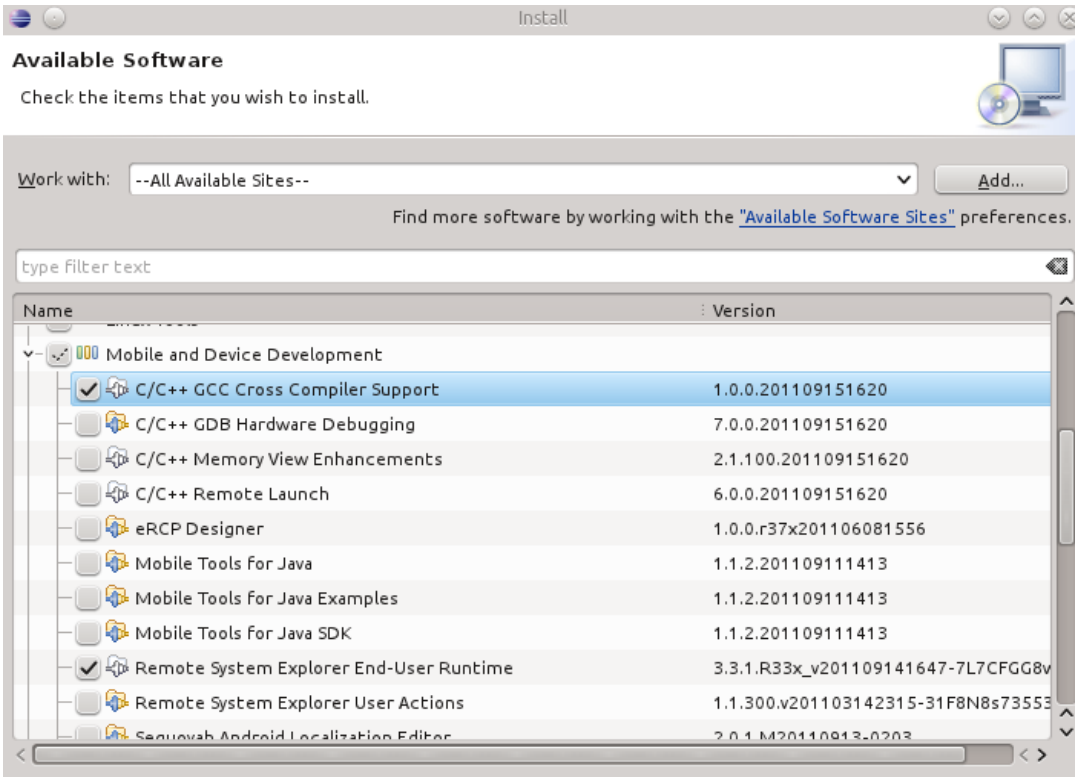
- DropBear or OpenSSH installed & connections can be established
- libstdcpp (optional for C++)
- openssh-sftp-server
- gdbserver

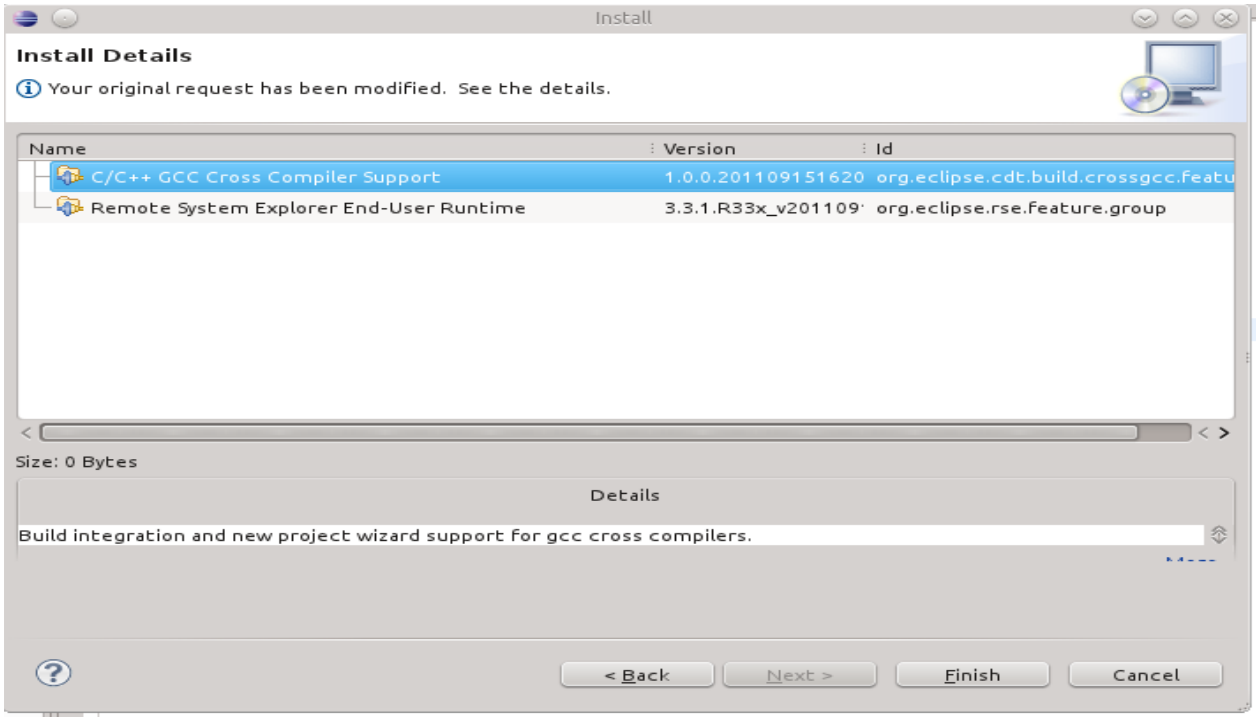
5 Eclipse Prerequisites

Download your desired Eclipse IDE for C/C++ Developers, extract archive, execute eclipse and enter workbench.

We have to install additional eclipse packages: enter *Menu → Help → Install new Software* ..

and select in section “Mobile and Device Development” packages “C/C++ GCC Cross Compiler Support” and “Remote System Explorer End-User Runtime”.

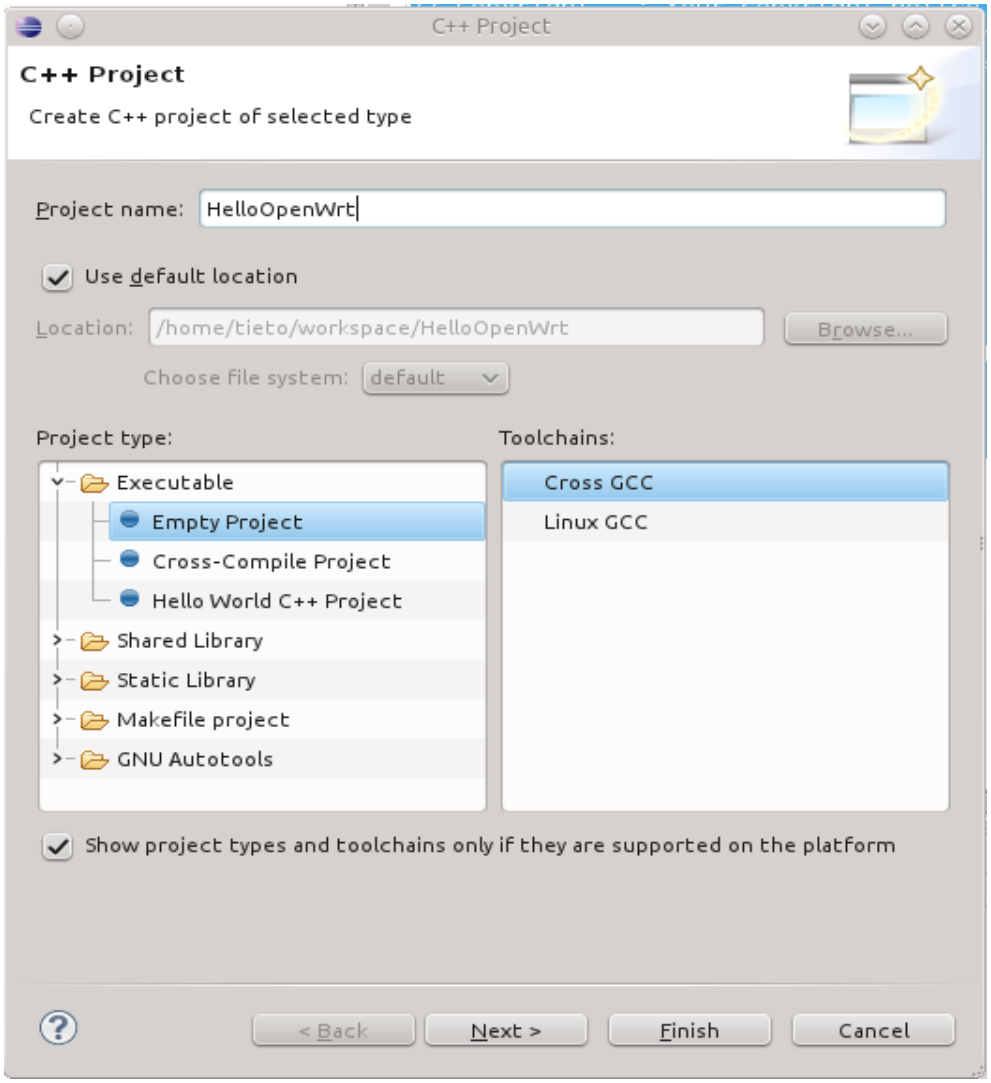




Finish and restart eclipse.

6 Eclipse Cross Compiler Project Setup

Create a new project: *Menu → File → New C++ Project (* resp. C project).



The next settings depends on your target device and where your buildroot has been installed.

We have to evaluate your specific target settings:

6.1 Tool command prefix (target specific)

```
tieto@vubuntu:~/openwrt/trunk$ find ./staging_dir -path "./staging_dir/toolchain*"  -name
*openwrt-linux
```

On my build system I get as result

```
./staging_dir/toolchain-mips_r2_gcc-4.5-linaro_uClibc-0.9.32/mips-openwrt-linux
./staging_dir/toolchain-i386_gcc-4.5-linaro_uClibc-0.9.32/i486-openwrt-linux
```

Hence for “Tool command prefix” we have to enter “*mips-openwrt-linux-*” for ar7xxx based resp. “*i486-openwrt-linux-*” for generic x86 / e.g. vmware target device.

6.2 Tool command path (target specific)

We can reuse the above finding results to get the required full path:

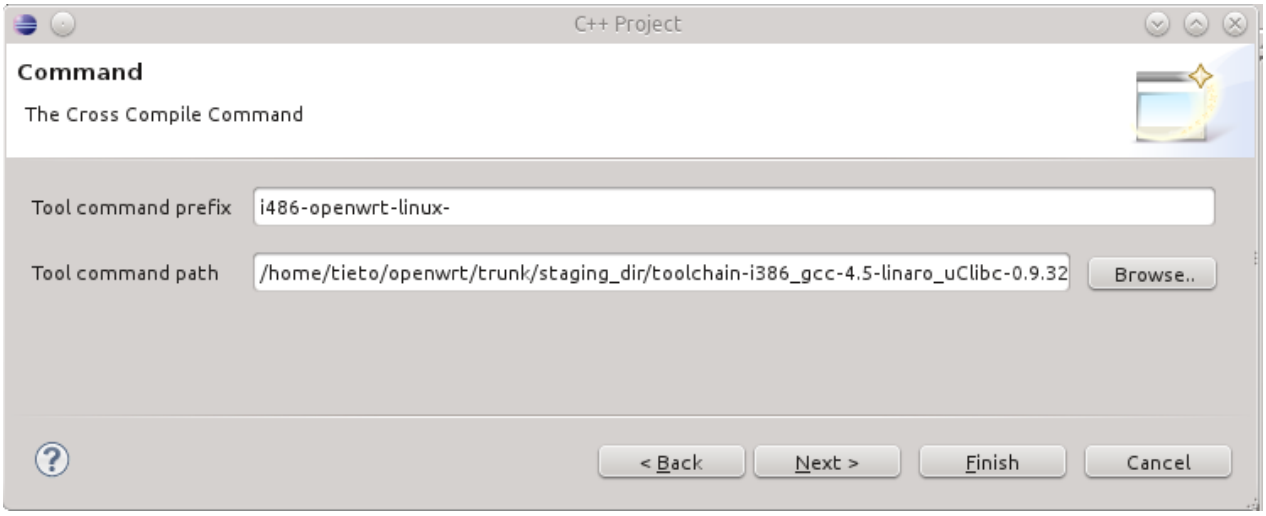
For above example ar7xxx target:

```
[YOUR_BR_HOME]/staging_dir/toolchain-mips_r2_gcc-4.5-linaro_uClibc-0.9.32
```

resp. for my generic x86 target:

```
[YOUR_BR_HOME]/staging_dir/toolchain-i386_gcc-4.5-linaro_uClibc-0.9.32
```

Don't forget to adapt these settings to YOUR specific build environment, COPY + PASTE from here may not work !



Finally enter your specific setting and press Finish button.

6.3 Cross Compile Hello World

Add src folder and source file

Menu → File → New Source Folder (src)

Menu → File → New Source File (src/HelloOpenwrt.cpp)

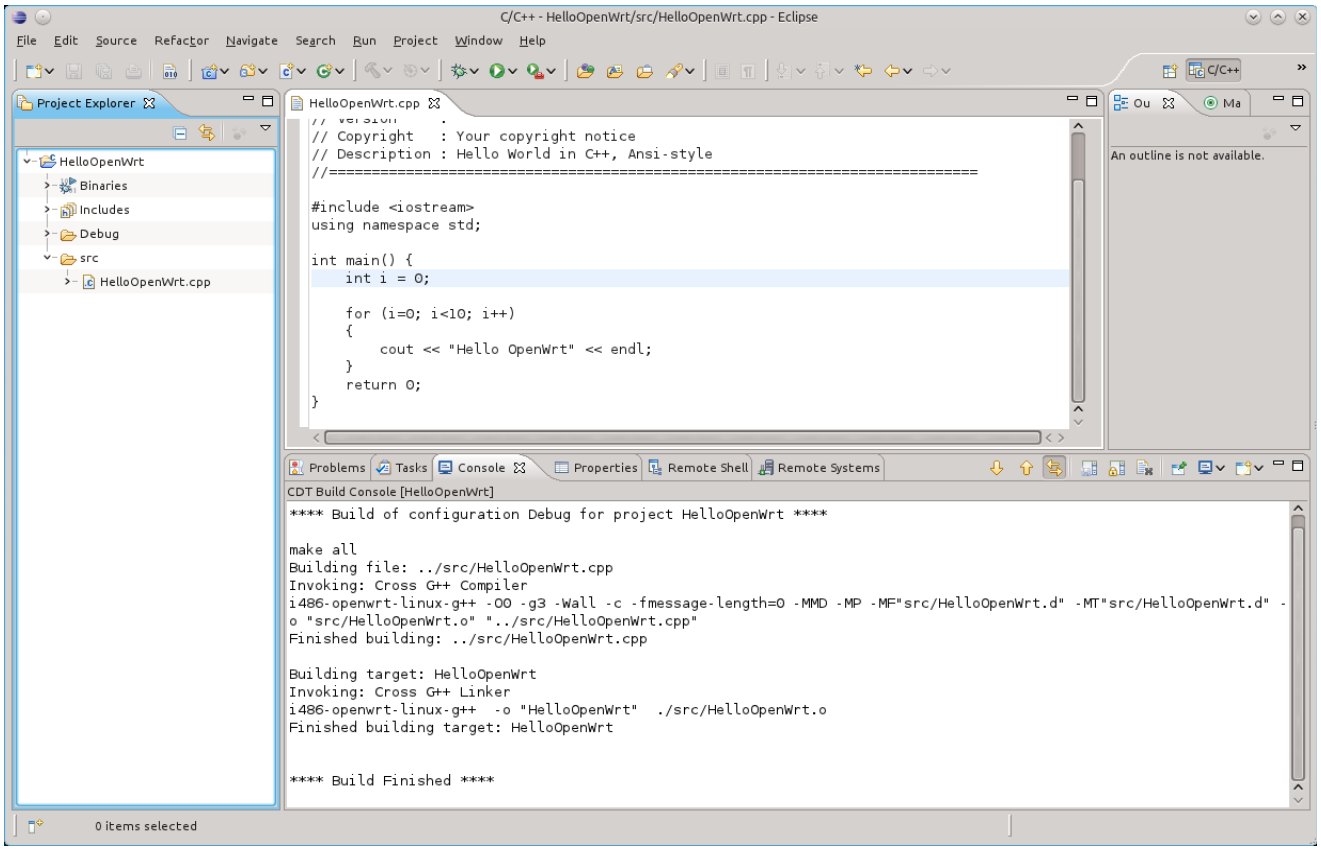
```
//============================================================================
// Name        : HelloOpenWrt.cpp
// Author      : irimi
// Version     :
// Copyright   : Your copyright notice
// Description : Hello World in C++, Ansi-style
//============================================================================

#include <iostream>
using namespace std;

int main() {
      int i = 0;

      for (i=0; i<10; i++)
      {
            cout << "Hello OpenWrt" << endl;
      }
      return 0;
}
```

If all was configured correctly you should be able to call *Menu->Project-Build all* without errors.

But you can't excute the created bin file on your build system, remember your binary is cross-compiled ☺.

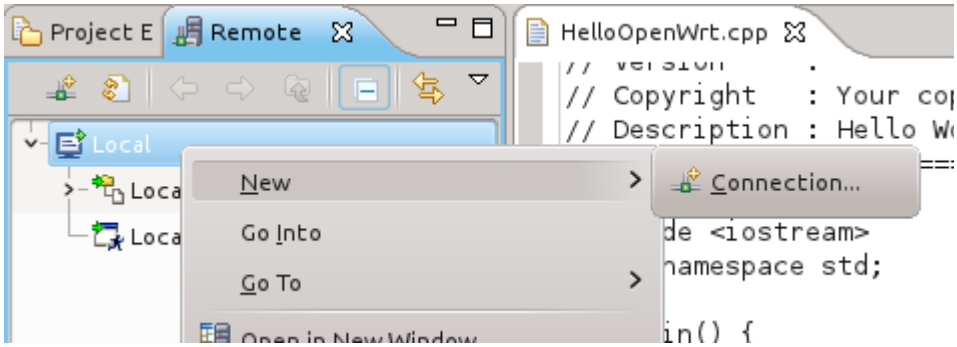
```
**** Build of configuration Debug for project HelloOpenWrt ****

make all
Building file: ../src/HelloOpenWrt.cpp
Invoking: Cross G++ Compiler
i486-openwrt-linux-g++ -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP
-MF"src/HelloOpenWrt.d" -MT"src/HelloOpenWrt.d" -o "src/HelloOpenWrt.o"
"../src/HelloOpenWrt.cpp"
Finished building: ../src/HelloOpenWrt.cpp

Building target: HelloOpenWrt
Invoking: Cross G++ Linker
i486-openwrt-linux-g++  -o "HelloOpenWrt"  ./src/HelloOpenWrt.o
Finished building target: HelloOpenWrt


**** Build Finished ****
```

Your Eclipse IDE will look like this:

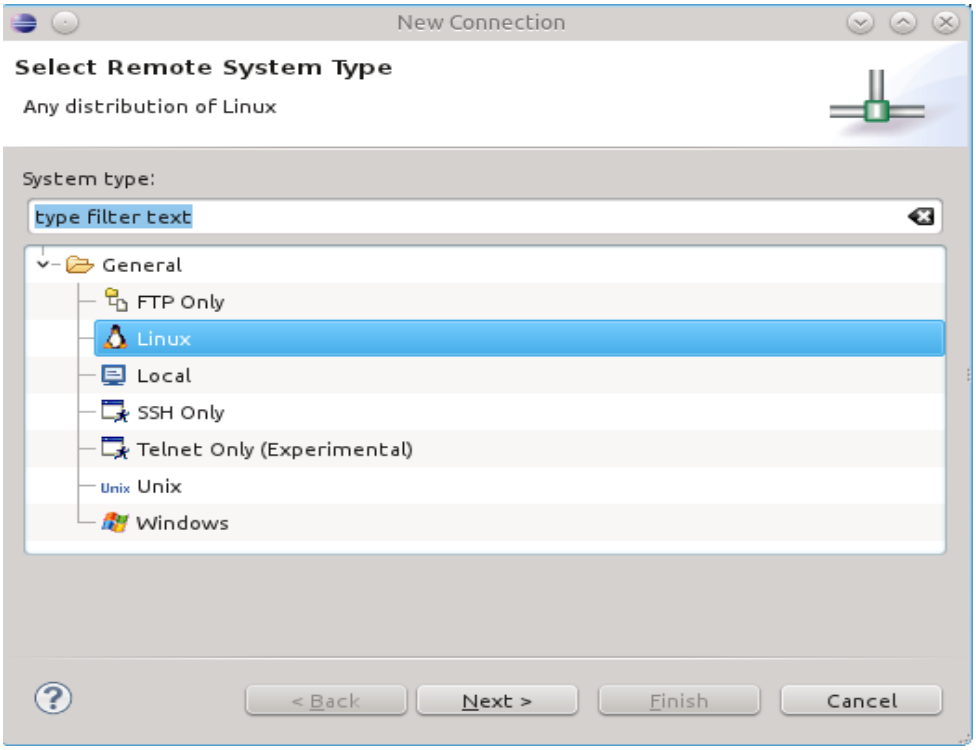


7 Remote Target Setup

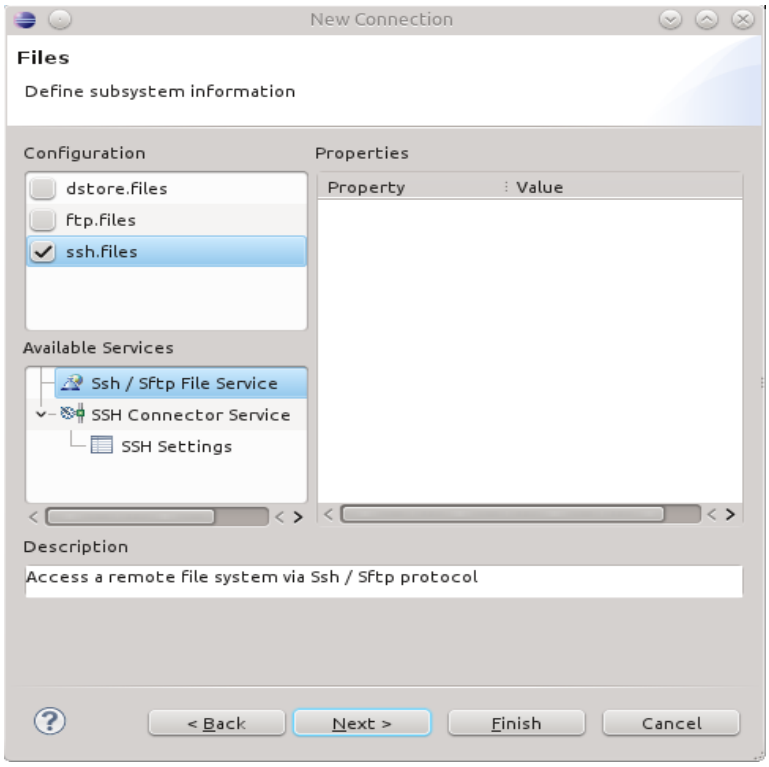
If you program was successfully compiled, you have to install it on your target device to execute it. At this point Eclipse will help with a nice feature: *Menu → Window → Show View → Remote System* and we create a new connection:



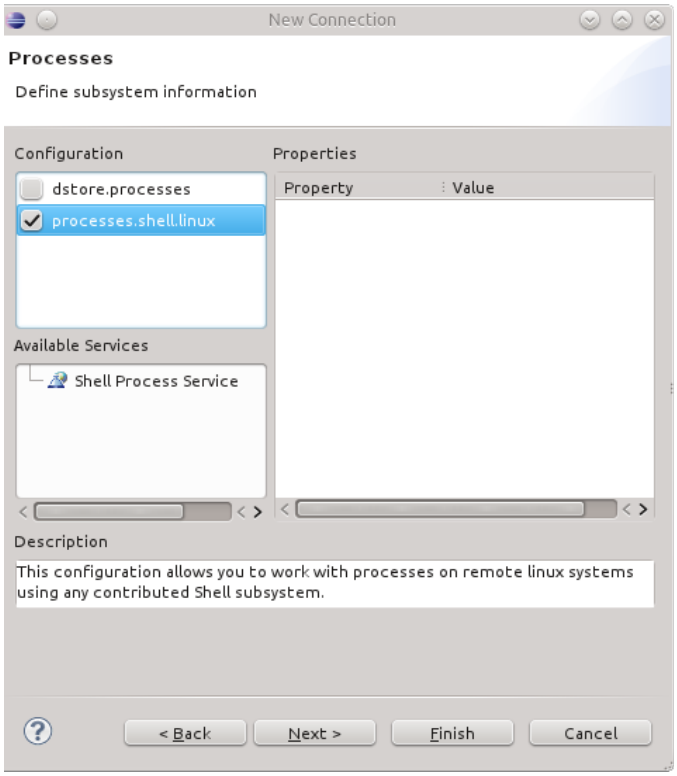
Now select Linux



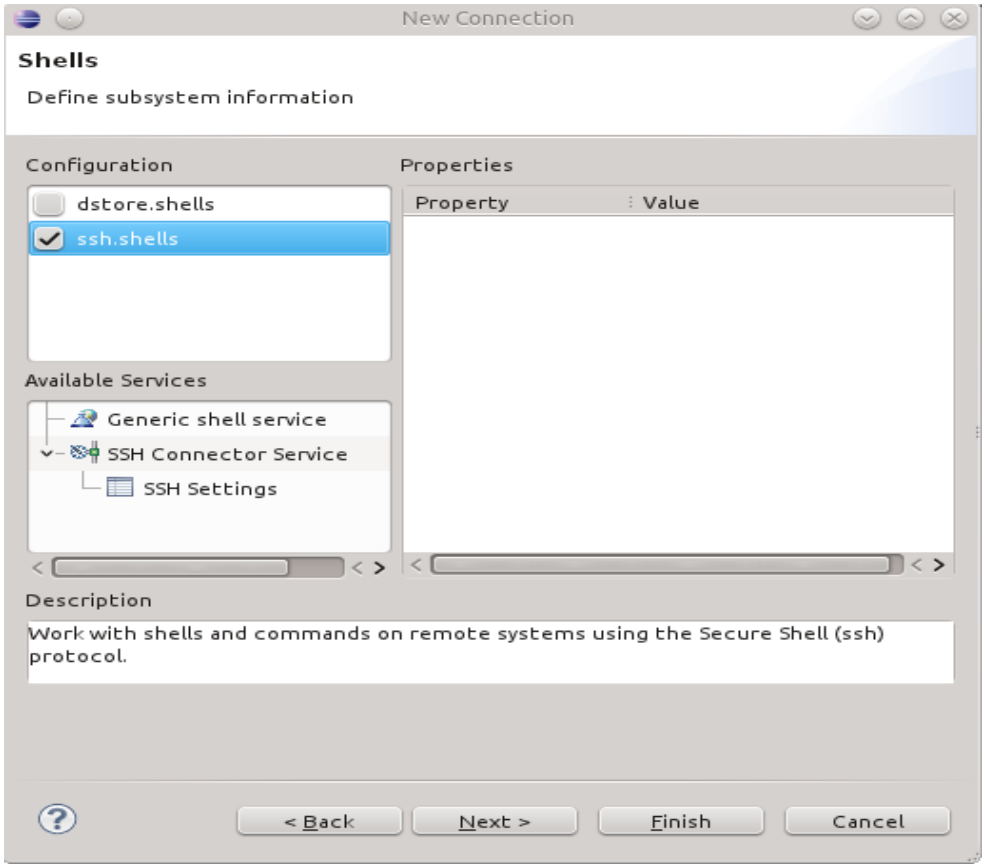
Now enter the target device's IP address resp. hostname and in next screen select .ssh files



Now select processes.shell.linux,



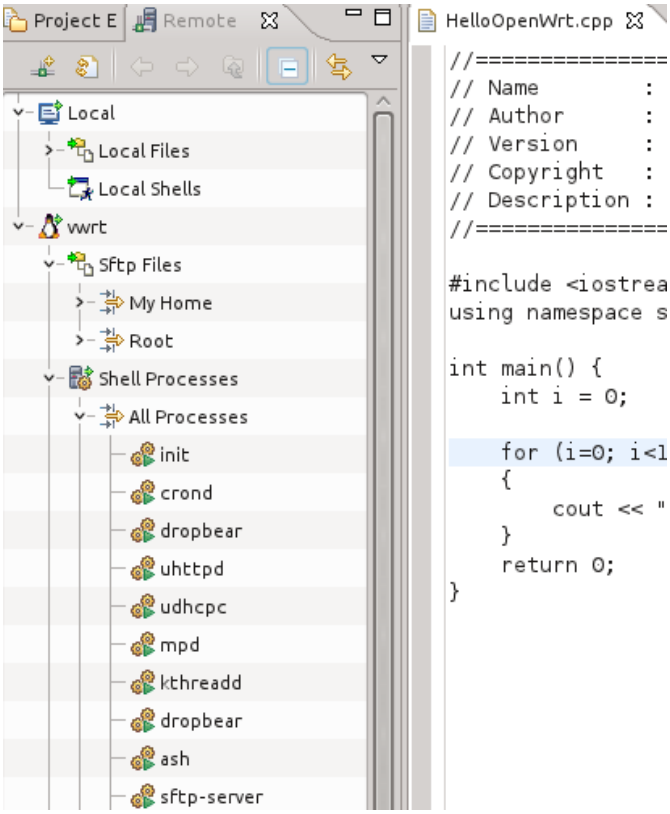
and select ssh.shells



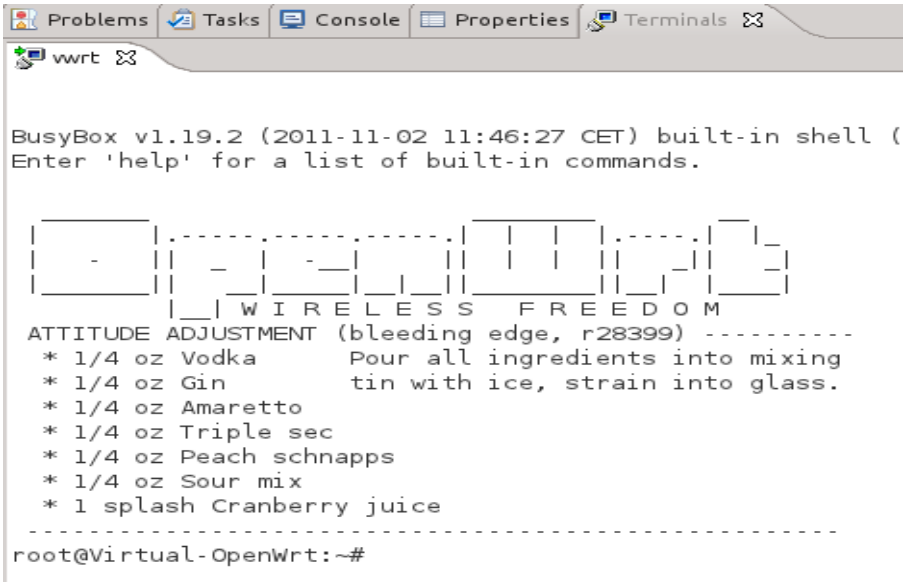
and finally Finish !

7.1 Browse Your Target Device

After entering user name + password you are now able to put files on/from your target devices via drag n' drop, you can even control the target's processes.



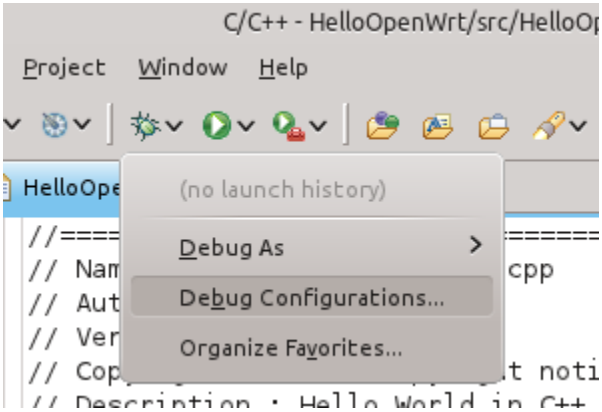
Or you can enter a ssh terminal in eclipse or copy/execute HelloOpenWrt bin file on your target device.



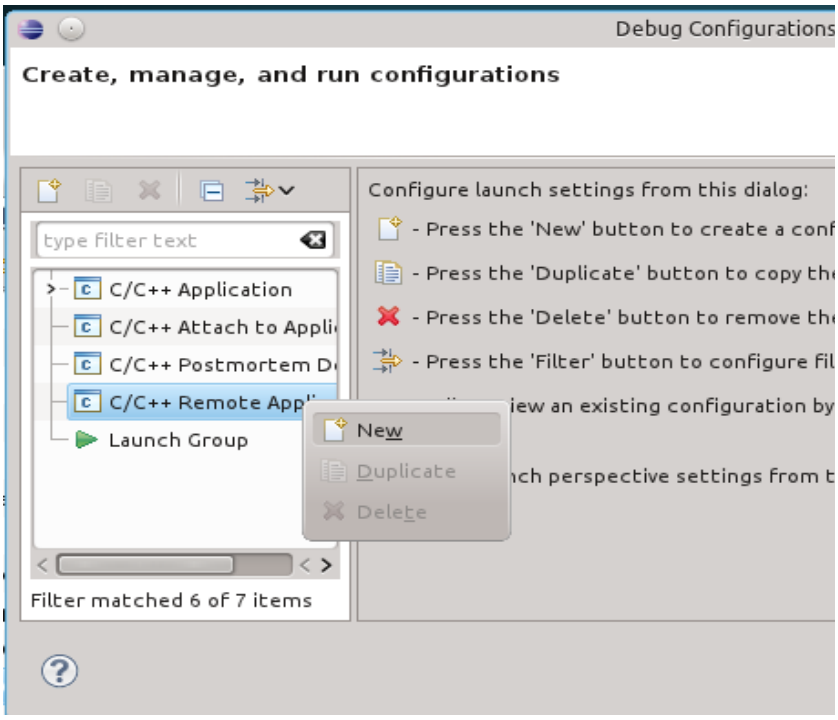
8 Remote gdb Debugger Setup

For target device remote debugging at first we have to define a debug configuration

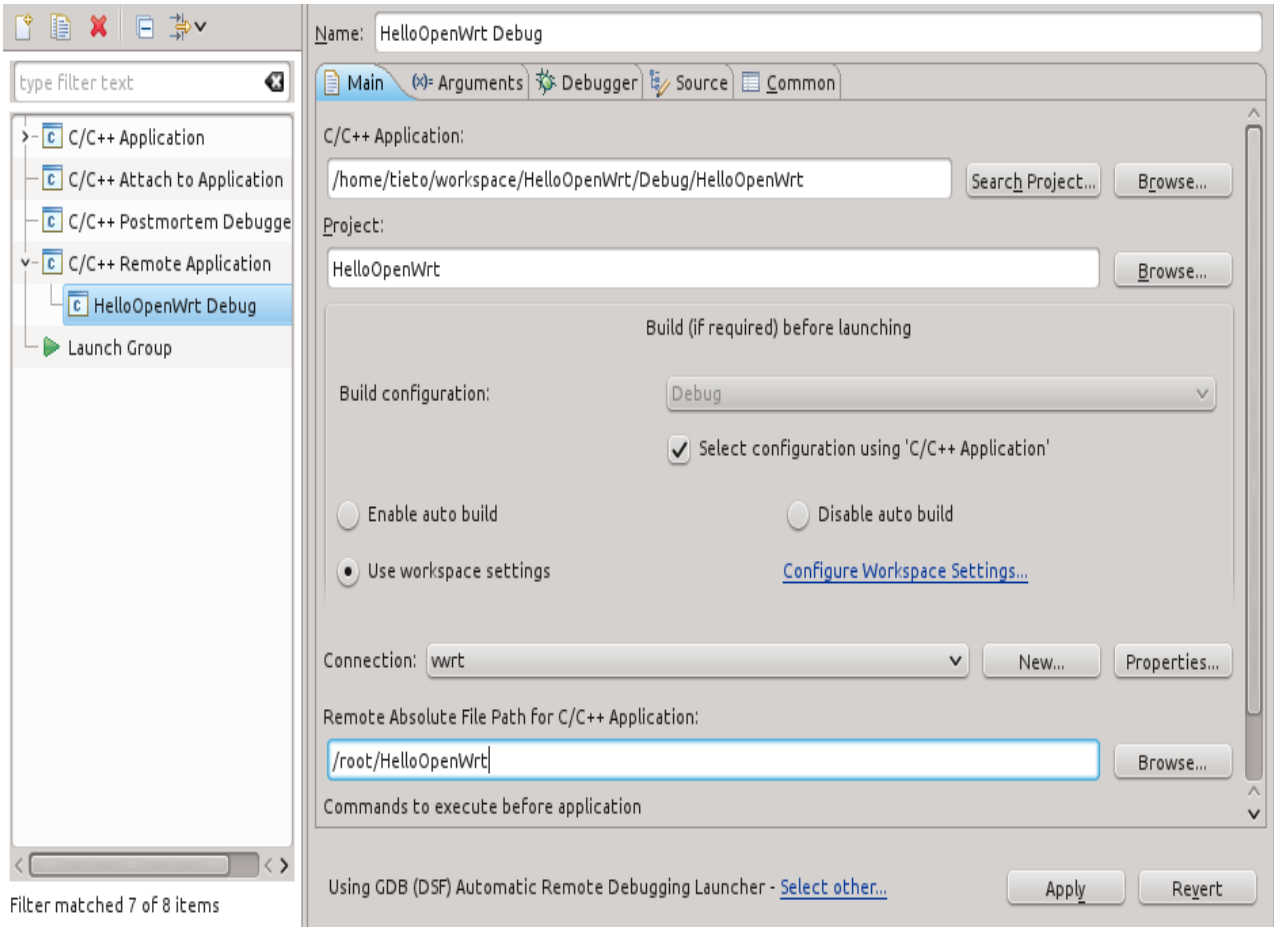
Left click on “bug”-button to enter “Debug Configurations”



and we create a new C/C++ Remote Application Debug Configuration



- In “Main” at C/C++ App adapt local file path to your application
- Change at “Connection” to your already defined target device remote connection (see chapter Remote Target Setup).
- don't forget to define the correct “Remote Absolute File Path for C/C++ Application”



Now click on “Debugger” settings to define the correct host gdb file.

8.1 Target specific host gdb

As host gdb we can't use the */usr/bin/gdb* provided e.g. by Ubuntu, we must use the gdb which has been built by our toolchain. As well as the tool command prefix, the location depends on your specific target settings and we evaluate it again.

It is located somewhere in *./build_dir*:

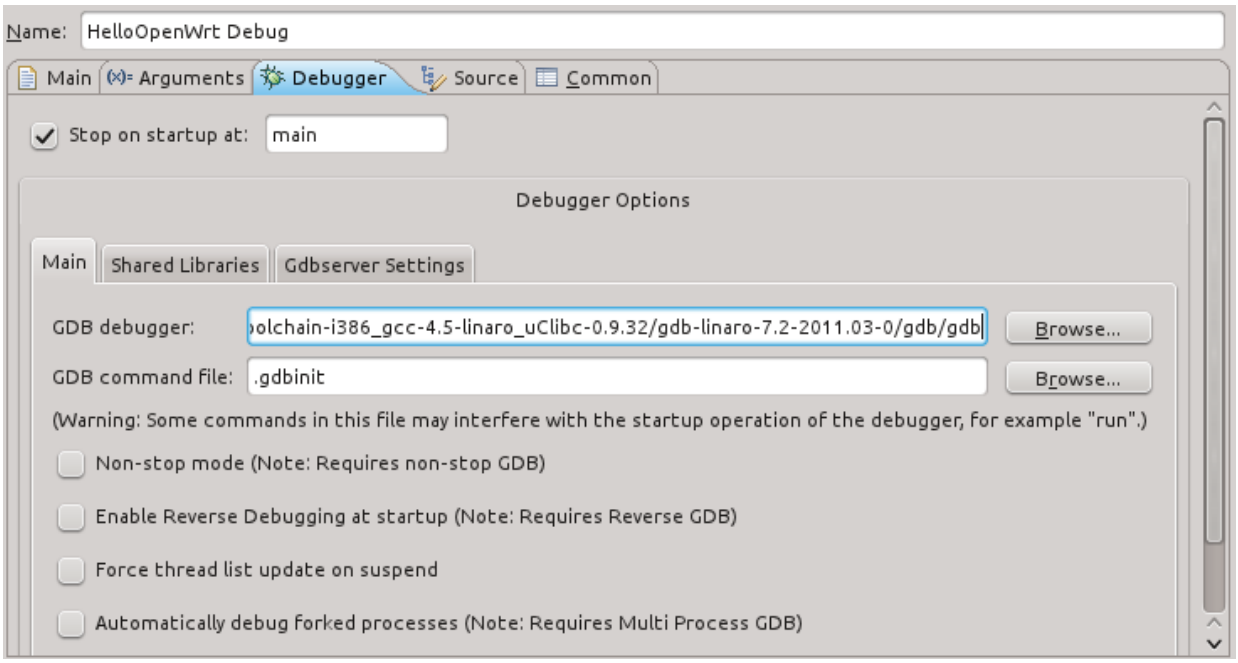
```
tieto@vubuntu:~/openwrt/trunk$ find ./build_dir -executable -type f -name gdb |grep toolchain
```

I got this result:

```
./build_dir/toolchain-i386_gcc-4.5-linaro_uClibc-0.9.32/gdb-linaro-7.2-2011.03-0/gdb/gdb
```

We have to enter the absolute file path at “GDB debugger”

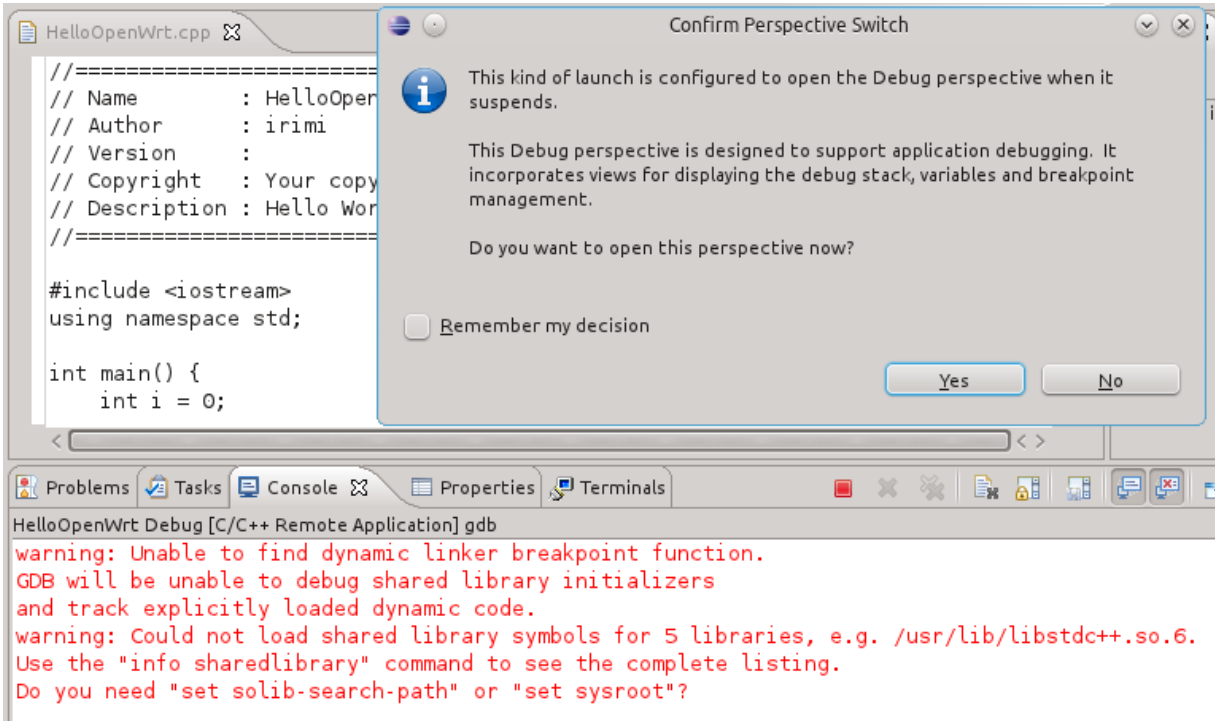
Remember these settings depend on YOUR specific build environment, COPY + PASTE from here may not work !!



Other changes are not required. Now you may press “Debug button” of settings window.

9 Remote Debugging Example

When you launch HelloOpenWrt Debug available at ”Bug-Button” the Debug View of eclipse will be opened.



The above warning can be ignored since you don't have enabled “Advanced configuration options (for developers)->Build Options->Debugging” in OpenWrt settings and rebuild all with debugging infos.

Congratulations, now you have a complete OpenWrt Development Suite !