Setup your PC for LibXenon Programming

This little How-To explains how to install the Xenon-Toolchain, LibXenonand configure the NetBeans IDE on Linux to develop with LibXenon for the Xbox360 Gaming Console.

Index

Requirements

1. Installed Linux

2. Basic Knowledge of Unix/Linux Environment

Installing Xenon-Toolchain

The Xenon-Toolchain is very essential when using LibXenon as it cross-compiles your code into powerpc-binaries which the Xenon-Hardware can execute!

To get the required files for this step you have to make sure that you got a GIT-Client and some other essential binaries installed first. You can install them on debian with the following command:

```
apt-get install libgmp3-dev libmpfr-dev libmpc-dev texinfo git-core gettext build-essential
```

Now you want to create a folder which will hold the Library sources, Xenon-Toolchain installation-files and other things from the Free60-Repository.

I would suggest putting these files into the **/home** directory, like this:

```
mkdir ~/free60project-github
```

```
cd ~/free60project-github
```

You can clone the Repository in the current directory now.

```
git clone git://github.com/Free60Project/libxenon.git
```

Now we create the target toolchain directory and chown to the current user (non-superuser), this gives us the freedom to do every further operation as non-superuser:

```
sudo mkdir /usr/local/xenon
```

```
sudo chown yourusername:yourusername -R /usr/local/xenon
```

** NOTE: yourusername is a placeholder for your normal username/useraccount*

Go into the toolchain directory now and start the actual Xenon-Toolchain Compile & Install.

```
cd libxenon/toolchain
```

```
./build-xenon-toolchain toolchain
```

This process will take, depending on your computer, approximately 2 hours.

Once it’s finished with compiling & installing it tells you to add two lines to `~/.bashrc` **- DON'T do it!**

They will usually look like this:

```
export DEVKITXENON="/usr/local/xenon"
export PATH="$PATH:$DEVKITXENON/bin:$DEVKITXENON/usr/bin"
```

Rather do it the following way:

```
sudo touch /etc/profile.d/devkitxenon.sh
sudo nano /etc/profile.d/devkitxenon.sh
```

Now, add the two lines to the (empty) file, press **CTRL+O** to save, press **ENTER** to confirm and finally **CTRL+X** to close the text editor.

You can test the installation now by **LOGGING OUT and back IN your current user**, opening a new terminal window and executing the following command:

```
xenon-gcc
```

If the output is: "**xenon-gcc: fatal error: no input files**", toolchain is installed and the paths are set correctly.

Installing LibXenon

Now, as the xenon-toolchain is set up properly, you can install the LibXenon Coding-Libraries.

At first navigate to the toolchain directory again.

```
cd ~/free60project-github/libxenon/toolchain
```

Now you can install LibXenon with one command

```
./build-xenon-toolchain libxenon
```

Once finished it should show:

```
Building libxenon...

libxenon installed successfully
```

To verfify the installation type:

```
./build-xenon-toolchain cube
```

On success it outputs:

```
Downloading Cube Sample

Building Cube Sample...

cube.elf32 compiled, run it via xell
```

If you like, you can rename "cube.elf32" to "xenon.elf" now and run it via XeLL.

Congratulations, the basics are done ☺

Installing NetBeans

Now you want to setup the actual Programming-IDE. Download NetBeans from

http://netbeans.org/

The C/C++ Edition is perfectly fine for our use. It's recommended to download the Version which includes Java-Runtime already so you don't have to care about it later on.

If you decided to take the NetBeans Installer which doesn't include it and you don't have JAVA SE installed yet, go to http://www.oracle.com/technetwork/java/javase/index.html, download the JDK Installer and follow the website's instructions to install it.

When you finished downloading the NetBeans-Installer you have to make it executable and execute it:

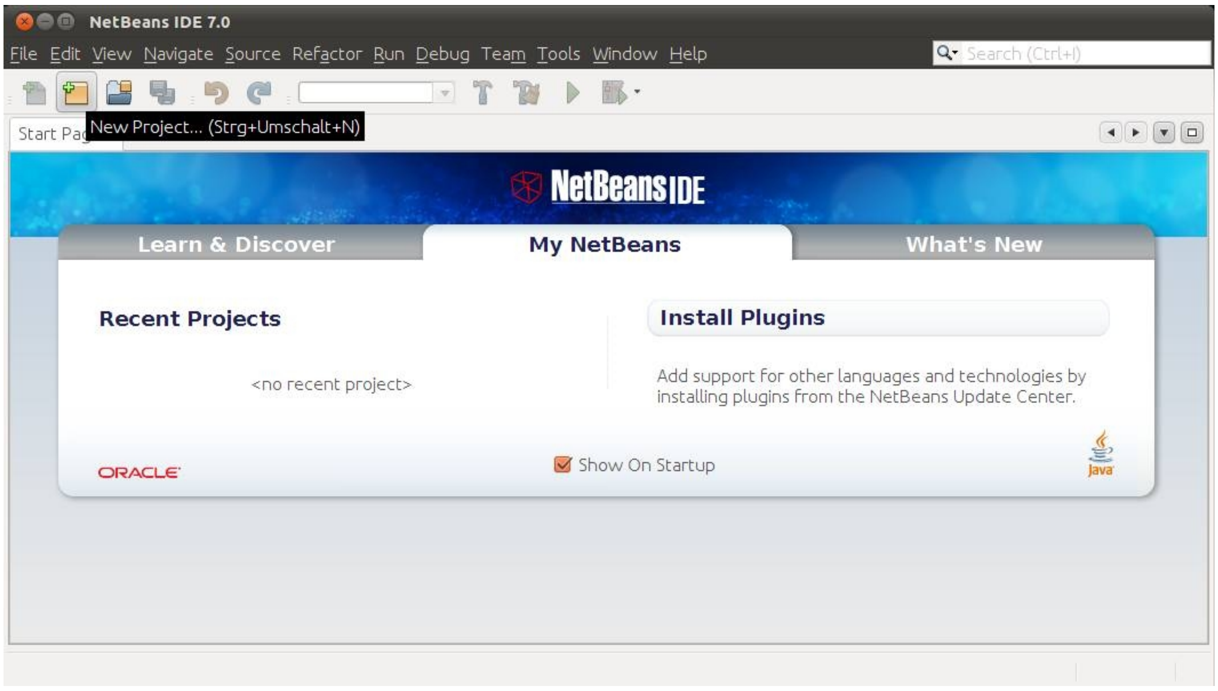
```
cd /wherever_the_installer_is_located/

chmod a+x netbeans-*-linux.sh

./netbeans-*-linux.sh
```

The Installation of NetBeans is pretty straight forward. When Netbeans is installed and started, proceed with “Configure NetBeans” which is the following step.

Configure NetBeans

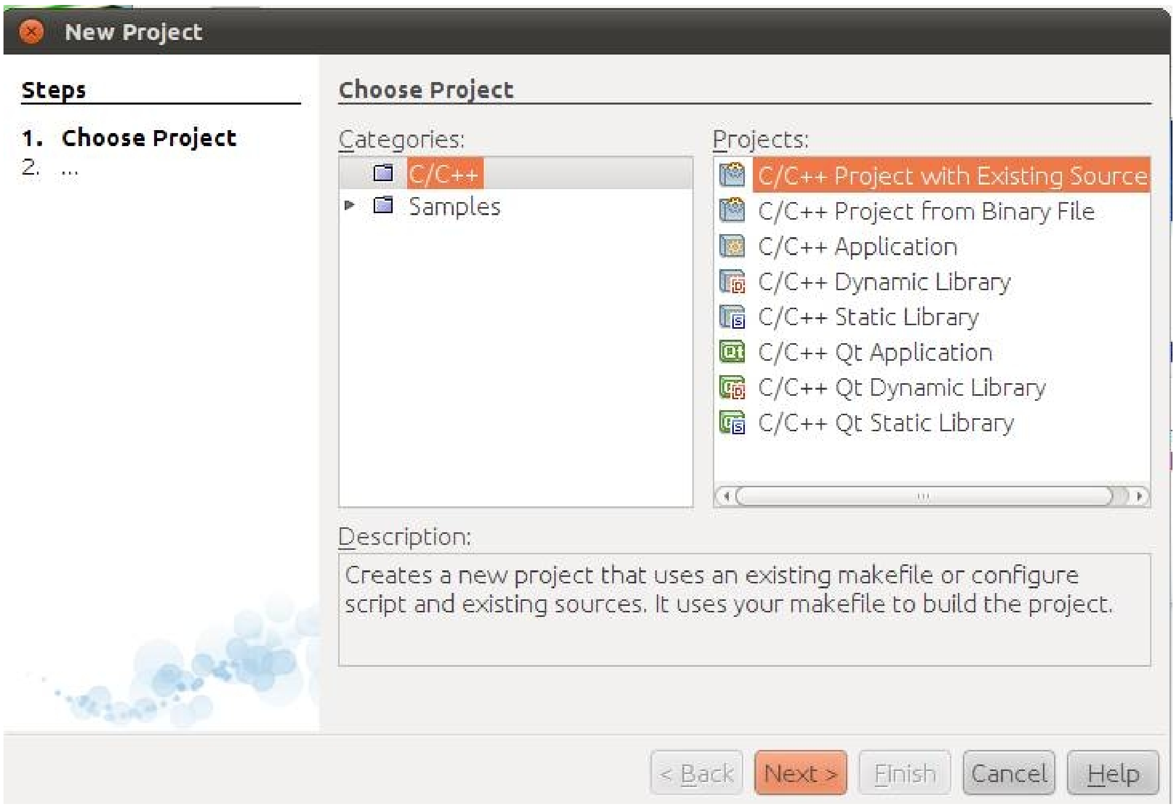
These are the basic steps how to configure a project to work with Code Assistance. Also it enables you to take a look at LibXenon functions all the time while coding.

NetBeans IDE and NetBeans Platform are based on software from netbeans.org, which has been dual licensed under the Common Development and Distribution License (CDDL) and the GNU General Public License version 2 with Classpath exception. For more information, please visit www.netbeans.org.



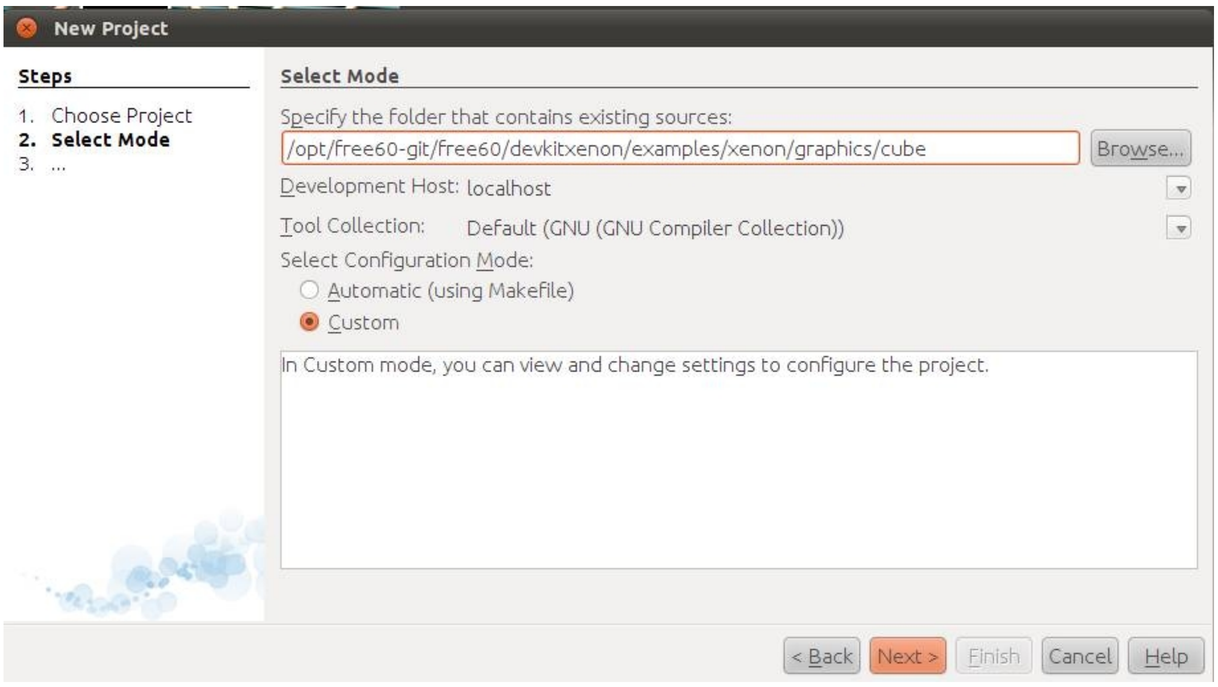
Add support for other languages and technologies by installing plugins from the NetBeans Update Center.

Click on “New Project”



Creates a new project that uses an existing makefile or configure script and existing sources. It uses your makefile to build the project.

Choose Category: “C/C++” and Projects: “C/C++ Project with Existing Sources”. Click “Next”.



Specify the folder that contains existing sources: /opt/free60-git/free60/devkitxenon/examples/xenon/graphics/cube

Development Host: localhost

Tool Collection: Default (GNU (GNU Compiler Collection))

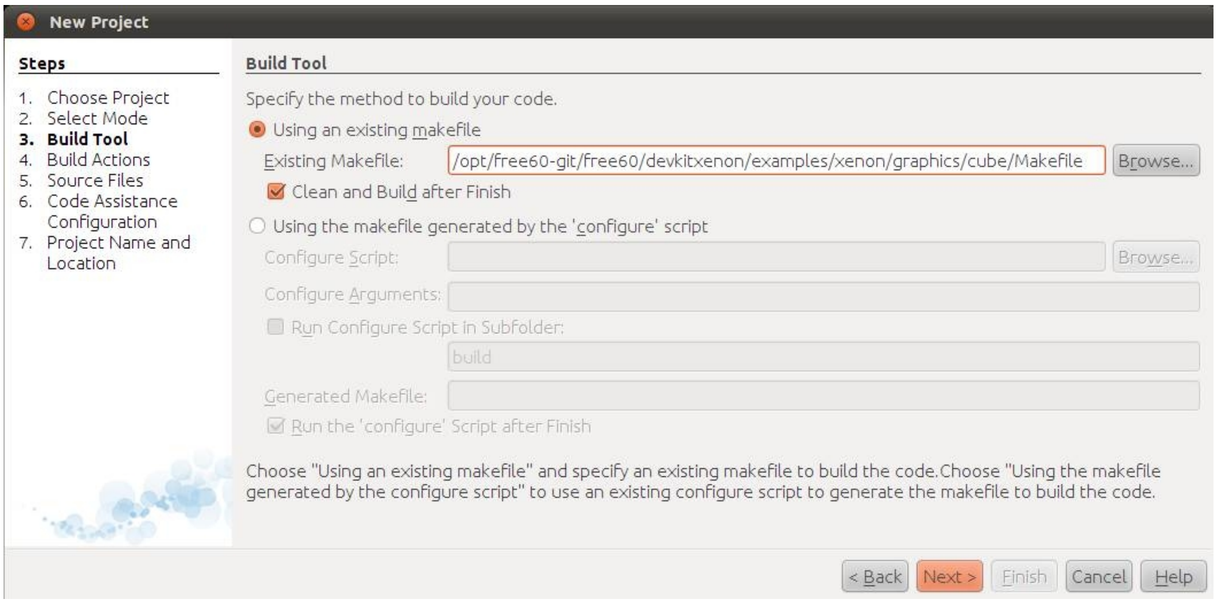
Select Configuration Mode:

- Automatic (using Makefile)
- [✓ Custom]

In Custom mode, you can view and change settings to configure the project.

Provide the path to the cube-sample. (for any new project its recommend to keep the cube folder structure or at least, keep the Makefile and modify it to your needs)

Choose “Custom” as the Configuration Mode. Click “Next”.


New Project

Steps

1. Choose Project
2. Select Mode
3. **Build Tool**
4. Build Actions
5. Source Files
6. Code Assistance Configuration
7. Project Name and Location

Build Tool

Specify the method to build your code.

[● Using an existing makefile]

Existing Makefile: /opt/free60-git/free60/devkitxenon/examples/xenon/graphics/cube/Makefile

[✓] Clean and Build after Finish

○ Using the makefile generated by the 'configure' script

Configure Script:

Configure Arguments:

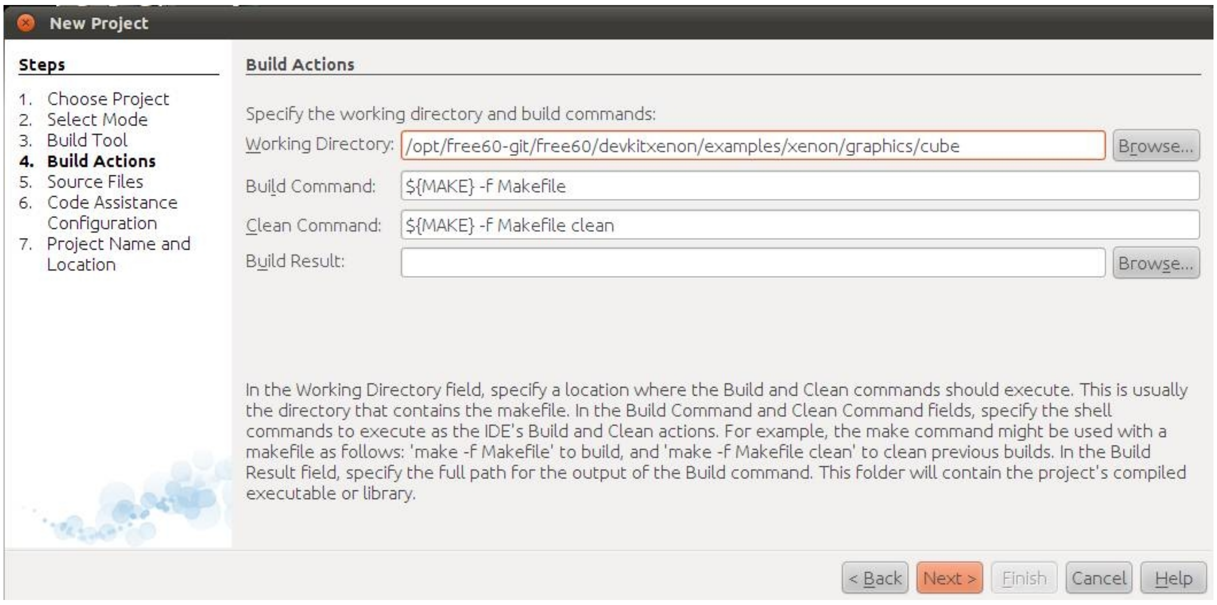
Run Configure Script in Subfolder: build

Generated Makefile:

[✓] Run the 'configure' Script after Finish

Choose "Using an existing makefile" and specify an existing makefile to build the code.Choose "Using the makefile generated by the configure script" to use an existing configure script to generate the makefile to build the code.

You can keep the “Build Tool”-options that NetBeans suggests as it already found the Makefile in the Project’s folder automaticly. If it doesn’t find it in future projects you create, you have to adjust the path in this step of course. Click “Next”.


New Project

Steps

1. Choose Project
2. Select Mode
3. Build Tool
4. **Build Actions**
5. Source Files
6. Code Assistance Configuration
7. Project Name and Location

Build Actions

Specify the working directory and build commands:

Working Directory: /opt/free60-git/free60/devkitxenon/examples/xenon/graphics/cube

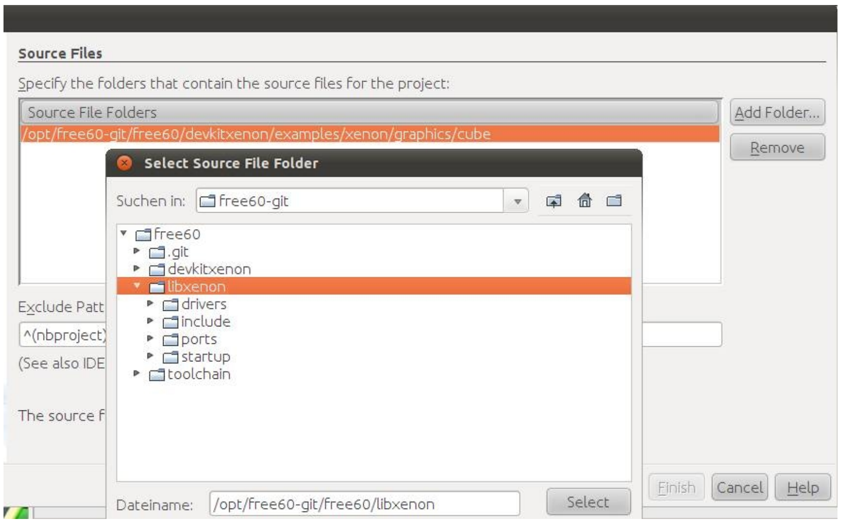
Build Command: ${MAKE} -f Makefile

Clean Command: ${MAKE} -f Makefile clean

Build Result:

In the Working Directory field, specify a location where the Build and Clean commands should execute. This is usually the directory that contains the makefile. In the Build Command and Clean Command fields, specify the shell commands to execute as the IDE's Build and Clean actions. For example, the make command might be used with a makefile as follows: 'make -f Makefile' to build, and 'make -f Makefile clean' to clean previous builds. In the Build Result field, specify the full path for the output of the Build command. This folder will contain the project's compiled executable or library.

The working directory parameters are also good like NetBeans suggests. Click “Next”.


Source Files

Specify the folders that contain the source files for the project:

Source File Folders
/opt/free60-git/free60/devkitxenon/examples/xenon/graphics/cube

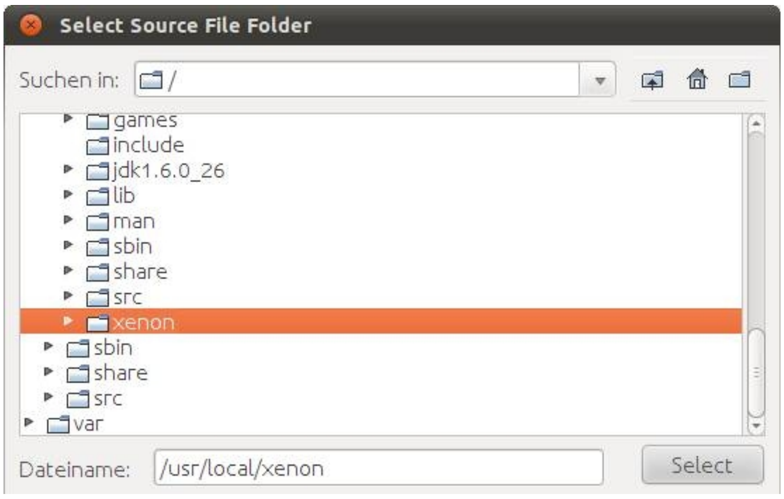
Select Source File Folder

Suchen in: free60-git

free60
.git
devkitxenon
libxenon
drivers
include
ports
startup
toolchain

Exclude Patt
^(nbproject)
(See also IDE
The source f

Dateiname: /opt/free60-git/free60/libxenon

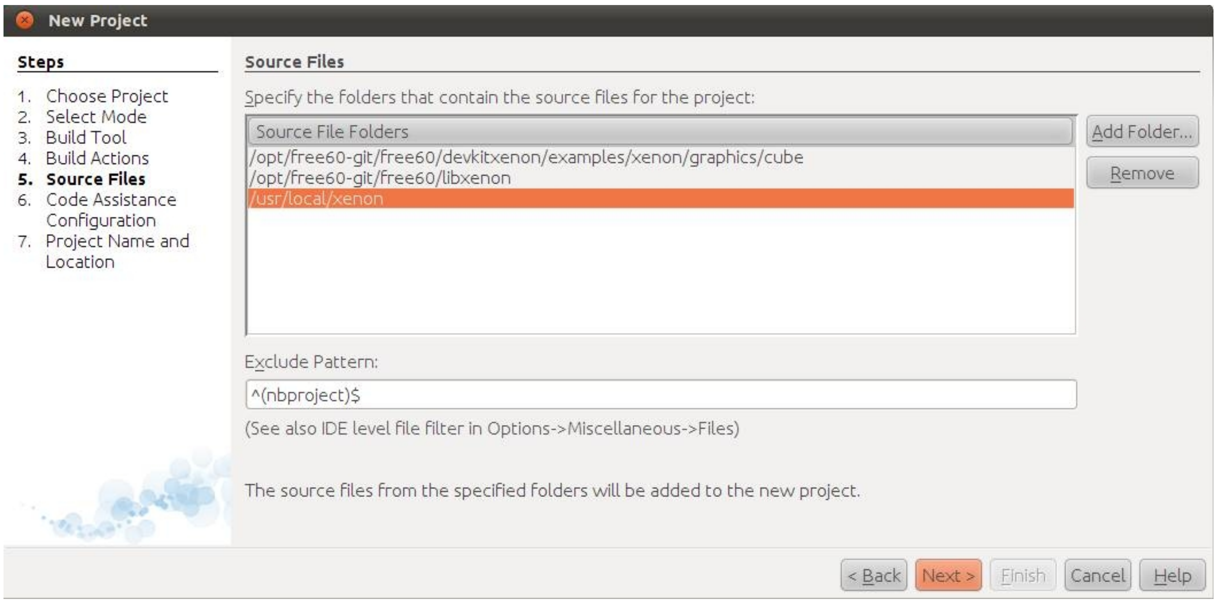

Select Source File Folder

Suchen in: /

games
include
jdk1.6.0_26
lib
man
sbin
share
src
xenon
sbin
share
src
var

Dateiname: /usr/local/xenon

In the “Source Files”-Options you have to point NetBeans to the folders you would like to see in your current project. It’s useful to add the LibXenon-directory: **`~/free60project-github/libxenon/libxenon`** too - to look up all libxenon power while working on your project.


New Project

Steps
1. Choose Project
2. Select Mode
3. Build Tool
4. Build Actions
5. **Source Files**
6. Code Assistance Configuration
7. Project Name and Location

Source Files

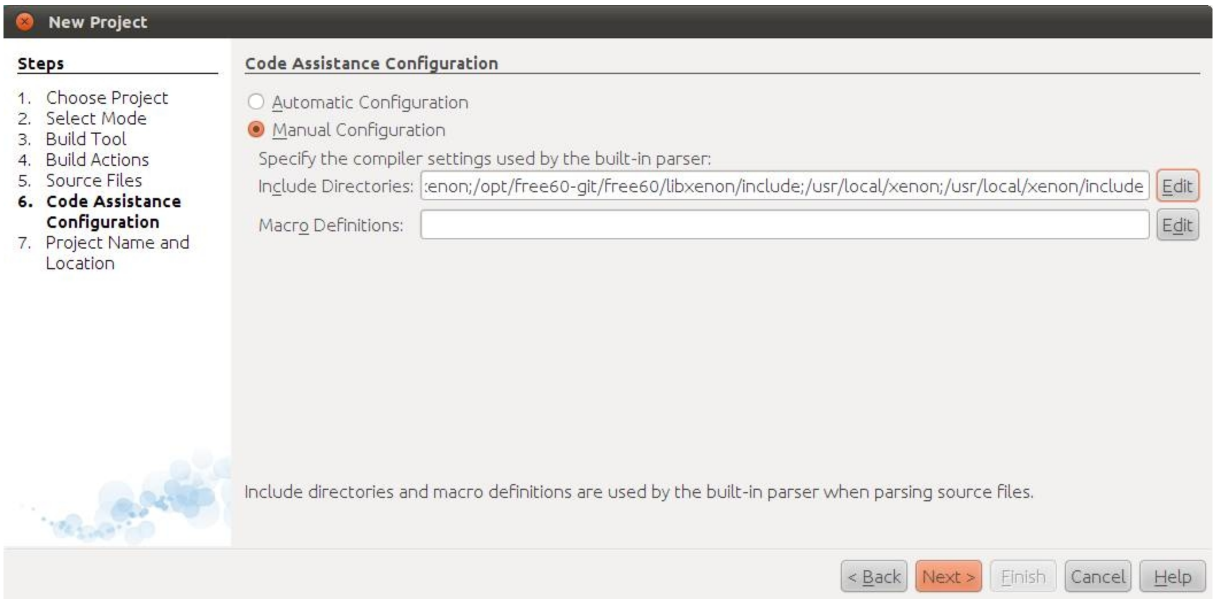
Specify the folders that contain the source files for the project:

Source File Folders
/opt/free60-git/free60/devkitxenon/examples/xenon/graphics/cube
/opt/free60-git/free60/libxenon
/usr/local/xenon

Exclude Pattern:
^(nbproject)$
(See also IDE level file filter in Options->Miscellaneous->Files)

The source files from the specified folders will be added to the new project.

The “Source Files”-Tab should look like that now. Click “Next”.


New Project

Steps

1. Choose Project
2. Select Mode
3. Build Tool
4. Build Actions
5. Source Files
6. **Code Assistance Configuration**
7. Project Name and Location

Code Assistance Configuration

○ Automatic Configuration
● Manual Configuration

Specify the compiler settings used by the built-in parser:

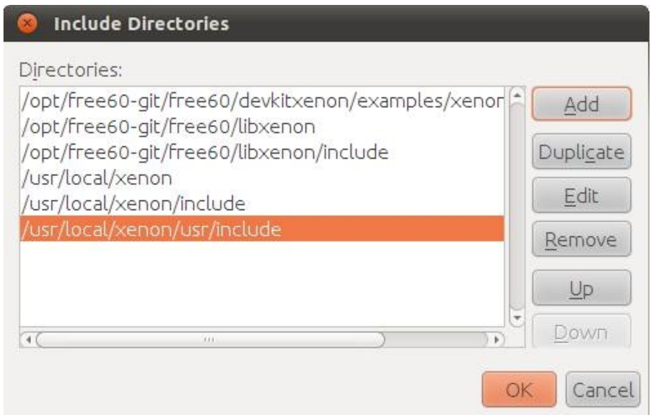
Include Directories: :enon;/opt/free60-git/free60/libxenon/include;/usr/local/xenon;/usr/local/xenon/include

Macro Definitions:

Include directories and macro definitions are used by the built-in parser when parsing source files.

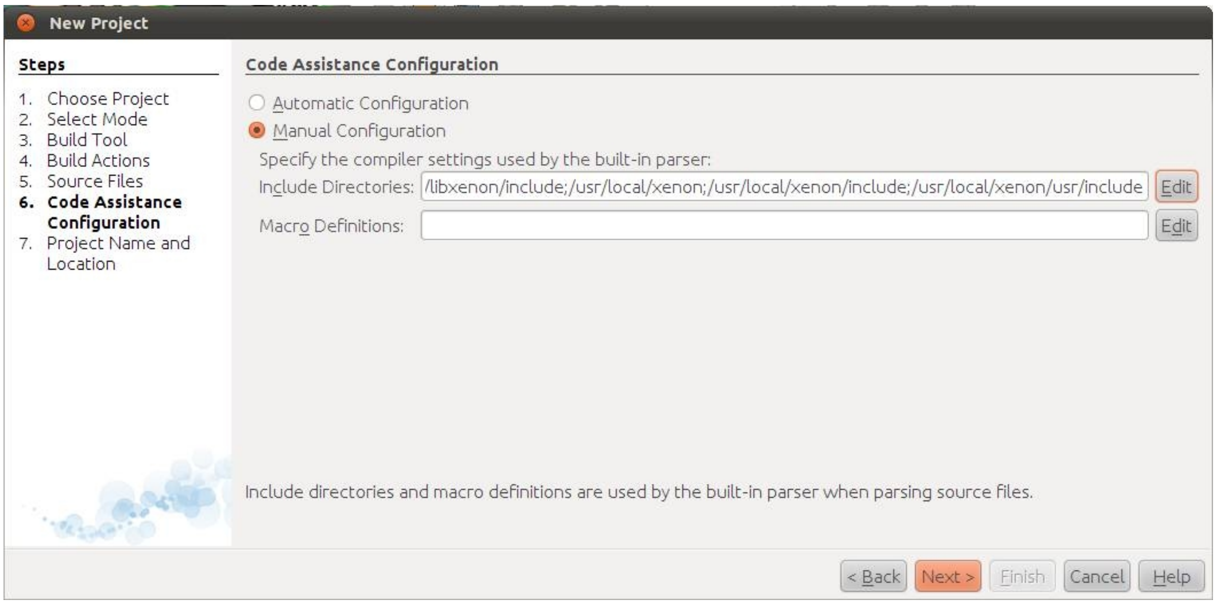
In this Tab it wants the Include-directories to configure the “Code Assistance”. It already got most of the paths correct as it took them from the prior “Source Code”-Tab.

You just have to add **`/usr/local/xenon/usr/include`** as seen on the following picture:


Include Directories

Directories:

```
/opt/free60-git/free60/devkitxenon/examples/xenon
/opt/free60-git/free60/libxenon
/opt/free60-git/free60/libxenon/include
/usr/local/xenon
/usr/local/xenon/include
/usr/local/xenon/usr/include
```



Code Assistance Configuration

Automatic Configuration [✓ Manual Configuration]

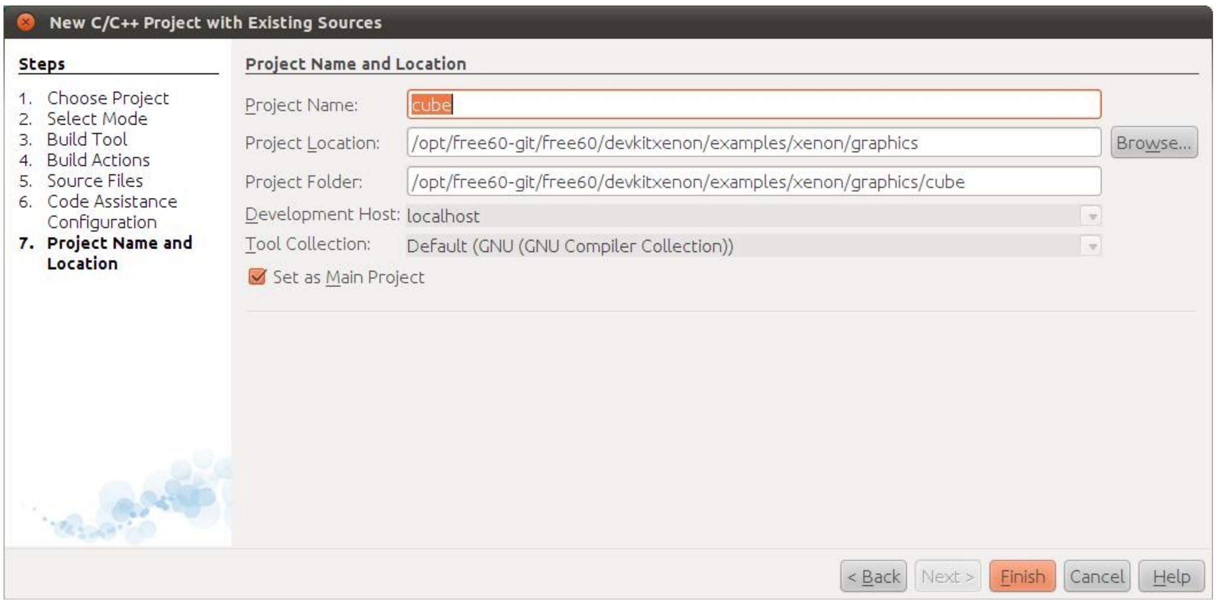
Specify the compiler settings used by the built-in parser:

Include Directories: /libxenon/include;/usr/local/xenon;/usr/local/xenon/include;/usr/local/xenon/usr/include

Macro Definitions:

Include directories and macro definitions are used by the built-in parser when parsing source files.

Click “Next”



Project Name and Location

Project Name: cube

Project Location: /opt/free60-git/free60/devkitxenon/examples/xenon/graphics

Project Folder: /opt/free60-git/free60/devkitxenon/examples/xenon/graphics/cube

Development Host: localhost

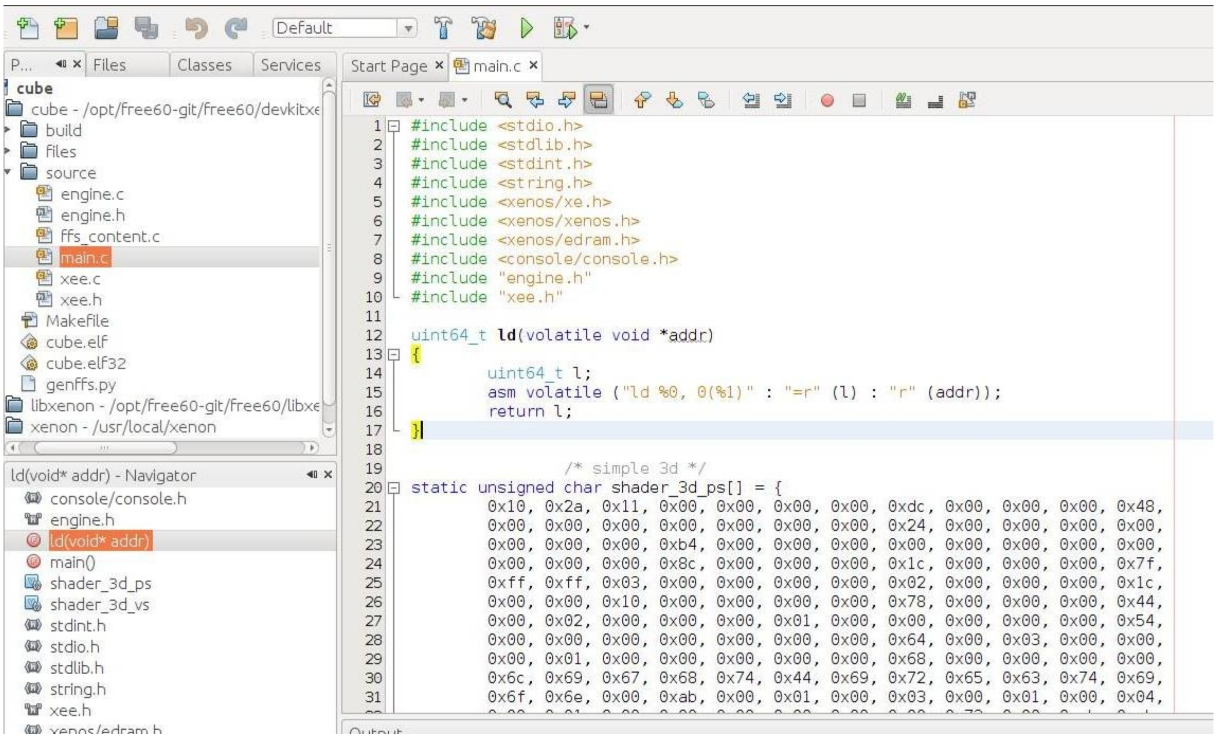
Tool Collection: Default (GNU (GNU Compiler Collection))

✓ Set as Main Project

If you like, you can adjust the project’s name now. Then click “Finish”.

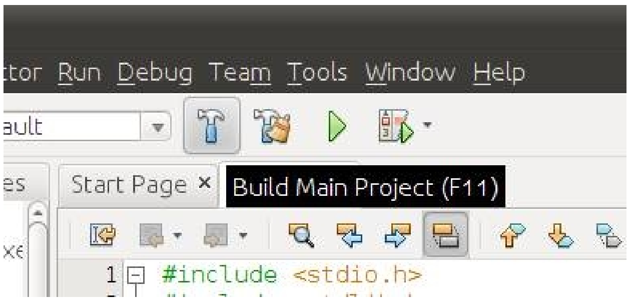
NetBeans loads the project now and parses the Include- and Source- Directories. It will take some time to finish.

On the left you see the Project Window with your main-project loaded + the chosen Source File Directories. Navigate into the main-project, open the source code and do what you like to do ☺

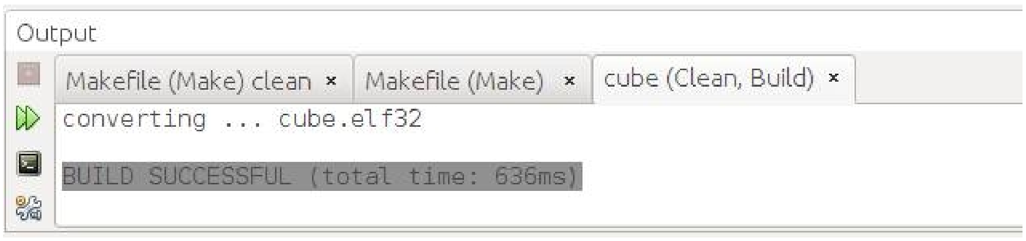


```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string.h>
#include <xenos/xe.h>
#include <xenos/xenos.h>
#include <xenos/edram.h>
#include <console/console.h>
#include "engine.h"
#include "xee.h"

uint64_t ld(volatile void *addr)
{
        uint64_t l;
        asm volatile ("ld %0, 0(%1)" : "=r" (l) : "r" (addr));
        return l;
}

                /* simple 3d */
static unsigned char shader_3d_ps[] = {
        0x10, 0x2a, 0x11, 0x00, 0x00, 0x00, 0x00, 0xdc, 0x00, 0x00, 0x00, 0x48,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0xb4, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x8c, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x7f,
        0xff, 0xff, 0x03, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x1c,
        0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x44,
        0x00, 0x02, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x54,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x64, 0x00, 0x03, 0x00, 0x00,
        0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x68, 0x00, 0x00, 0x00, 0x00,
        0x6c, 0x69, 0x67, 0x68, 0x74, 0x44, 0x69, 0x72, 0x65, 0x63, 0x74, 0x69,
        0x6f, 0x6e, 0x00, 0xab, 0x00, 0x01, 0x00, 0x03, 0x00, 0x01, 0x00, 0x04,
```

Once you are done with editing press the “Build” or “Clean & Build” button:



If everything goes well it will report the following in the output-window:



```
converting ... cube.elf32

BUILD SUCCESSFUL (total time: 636ms)
```

That’s it ;)

V1.1