Maemo Diablo Software Development Process for maemo SDK

Training Material

February 9, 2009

Contents

Chapter 1

Software Development Process for maemo SDK

Software development for the maemo is possible using many different programming languages. This chapter describes the process and common practises of developing software for the maemo platform. It also introduces the tools that form the maemo SDK environment.

1.1 Overview of the software development process using the maemo SDK environment

The development environment used in the process is called `maemo SDK` which is freely downloadable from the maemo community website maemo.org. The maemo SDK utilises `Scratchbox`, which is a cross-compilation toolkit and a "sandbox", designed for embedded Linux application development. Scratchbox is downloadable from its' website scratchbox.org.

The maemo SDK provides a development environment for creating software to Internet Tablets using a Linux desktop computer. The SDK runs inside the Scratchbox and contains all necessary compilers, tools, libraries and headers to develop software for the two target hardware architectures, Intel (`x86`) and ARMEL. Application development and preliminary testing of the software is done in x86 environment, which also includes the Hildon Desktop for running the applications on the desktop computer (using virtual X server, such as `Xephyr`) like they would run on the actual Internet Tablet. Using desktop computer for development makes the application development quite similar to normal Linux application development. Maemo SDK also has a support-plug-in for Eclipse, Integrated Development Environment (IDE) which speeds up and helps the development process radically.

After the preliminary testing on desktop computer is finished, the next phase is to cross-compile and package the application for the ARMEL architecture using the ARMEL target of the Scratchbox, and the application is ready to be run and tested on the Internet Tablet. Testing-phase using the actual Internet Tablet is important even when the application runs fine on the desktop environment as the SDK is not exactly 100% identical to the device.

Development tools and resources used in maemo application development process:

Scratchbox Scratchbox is a cross compilation toolkit designed to make embedded Linux application development easier. It also provides a full set of tools to integrate and cross compile an entire Linux distribution. The toolkit supports ARM architecture and x86 and few more are under development. Scratchbox supports multiple configurations for each developer in the same host machine.

maemo SDK rootstraps The rootstrap is a target root filesystem image for Scratchbox that can be used as a basis for development. Maemo SDK provides rootstraps for both x86 and ARMEL development.

Nokia binaries Software packages that are not available as a source code but may provide public API, like the contact information import/export library, GPS (location) libraries, address-book and presence-information libraries.

maemo tools Several tools for power users requiring more sophisticated development tools than provided in the standard maemo SDK package. Includes tools for code analysis, debugging, resource usage, test automation etc.

maemo.org repositories maemo.org website has a lot of different repositories that are meant to be used with standard Debian package installation tools. Different repositories offer different software, tools, source code etc. for the developers.

maemo.org documentation Documentation for maemo software development include HOW-TOs, tutorials, API References, manual pages and several other guides, available from the maemo.org website.

maemo examples Maemo examples package includes demonstrative source code for using different APIs and can be fetched from the maemo repositories.

Phases of software development process using C or C++ language:

1. Create project (possibly using templates) for the application
2. Create or update the source code and needed resources
3. Create or update the UI schema (possibly using UI builder)
4. Build the application with `x86 rootstrap`
5. Launch and test the application on `x86 rootstrap`
6. Debug the application on `x86 rootstrap`
7. Cross-compile the application with `ARMEL rootstrap`
8. Launch and test the application on the Internet Tablet

9. Debug application on Internet Tablet
10. Create ARMEL installation package for the Internet Tablet
11. Install ARMEL application package to the Internet Tablet

Phases of software development process using Python (and other script-languages):

1. Create project (possibly using templates) for the application
2. Create or update the source code and needed resources
3. Launch and test the application on `x86 rootstrap`
4. Debug the application on `x86 rootstrap`
5. Launch and test the application on the Internet Tablet
6. Debug application on the Internet Tablet
7. Create ARMEL installation package for the Internet Tablet
8. Install ARMEL application package to the Internet Tablet

Following chapters take a deeper look into the phases of the development process.

1.2 Creating project for application

Creating the project for an application can be done either from scratch, or by using several examples available from maemo web site as templates. Source files can be edited using your favourite text editor. Scratchbox creates a "sandbox", a separate filesystem (called `rootstrap`) under your normal filesystem, so it is advisable to create a symbolic link to Scratchbox folders for easier access of files from your desktop environment. Of course, using a console-based text editor (such as `nano`) inside Scratchbox shell is also possible.

As maemo uses Debian-based package management system for applications, it is a good practise to take that into account already when creating a project and create necessary files for packaging or use helper-applications for creating them, such as GNU Autotools.

Notice, the Hildon Desktop must be started before running the maemo applications inside the SDK.

The official programming language for maemo application development is C, but several other languages can be used with maemo also, for example C++ and Python.

There exists several UI-builder applications to speed up the creation of UI schema, most commonly used are Gazpacho and Glade.

Complete guide of setting up and using the development environment can be found from maemo Getting Started and maemo Application Development materials.

1.3 Building and running applications

When source code has been created with necessary resource-files, the application is ready for compiling and testing. The SDK provides all the usual Linux development tools inside the Scratchbox as well as the maemo application framework so the applications look and behave like they would on the Internet Tablet.

List of the most commonly used development tools provided by the `maemo rootstrap`:

GNU toolchain An umbrella term used of the programming tools produced by GNU Project, including:

- **GCC (GNU Compiler Collection)** Compilers and linkers for C, C++ etc.
- **GNU Autotools** Suite of programming tools designed to assist in `Makefile` generation and portability-issues.
- **GNU Make** Make is a tool which controls the generation of executables and other non-source files of a program from the program's source files.
- **GNU Binutils** A collection of programming tools for the manipulation of object code in various object file formats.

pkg-config Pkg-config is a helper tool used when compiling applications and libraries which helps you to insert the correct compiler options to find libraries.

Debian packaging tools Tools to create Debian software packages.

Process:

- Build the application using x86 target
- Launch the application on x86 target
  There is a helper shell-script on maemo rootstrap called `run-standalone.sh` which must be used when launching applications in scratchbox. The script sets the correct environment for the application to use the maemo application framework
- Test the application
- If needed, modify the source code and repeat

Debugging on x86 rootstrap

- Use the x86 target of the Scratchbox
- Launch the application in debugger
- Run and debug the application
- Modify and re-compile if needed

Debugging the application in x86 target can be done with regular Linux development and debugging tools, like GNU Debugger (`gdb`), valgrind, ltrace and strace. Some tools also provide a graphical user interface to be used for debugging.

Notice also that the `valgrind` tool is not available in ARMEL target, only in x86 target. Valgrind is a really powerful tool for memory leak detection, profiling etc.

If the debugging tool of your choice is missing, as the Scratchbox is practically a full Linux system, it is also possible to add a tool into it by compiling the tool for Scratchbox from the source code.

Other possibility is to run the application on Internet Tablet and debug remotely using `gdbserver` on Internet Tablet and `gdb` on PC environment, using either cable or wireless connection.

For more information about debugging, there is a comprehensive debugging guide on maemo.org website.

1.4 Cross-compiling for ARMEL

Cross-compiling the application for the Internet Tablet is really straightforward. Activate `ARMEL` target of Scratchbox and re-compile the application. There is no difference in the process of compiling the application to x86 or ARMEL targets. After compiling, your application binary is ready for the Internet Tablet architecture.

The ARMEL target should only be used for cross-compiling and packaging the applications for the Internet Tablet device, not for running and testing as the ARM CPU emulation of Qemu may not provide accurate enough emulation to finalise testing only on PC.

1.5 Running, testing and debugging applications on the Internet Tablet

Even though the SDK is quite accurately identical to the target environment of the Internet Tablet, it isn't 100% identical. Especially if your application is using some special hardware of the Internet Tablet, the application can behave differently in SDK than on the device.

Fortunately, testing the cross-compiled binary transparently on the Internet Tablet has been made possible from Scratchbox, using either SSH or a CPU transparency tool called `sbrsh` (Scratchbox Remote Shell). Connection to the Internet Tablet can be handled either by USB cable or wirelessly.

CPU transparency is a technique where ARMEL binaries are copied from the Linux PC side over the connection to the device where the binary is then executed natively by the Internet Tablets own ARM CPU. This is a handy way to test your ARMEL-binaries in a native ARM CPU instead of running them in Linux PC under the QEMU emulator. The QEMU emulator may not be identical with the real ARM device so using CPU transparency with the real device is a convenient way to test drive your ARMEL application. The graphical environment is still the Scratchbox (and virtual X server, Xephyr) environment.

- Launch the application transparently on the Internet Tablet
- Test the application
- If needed, modify and re-compile the source code and repeat

Check the maemo.org documentation for more information about CPU transparency.

It is possible to use `gdb` debugger remotely with a `gdbserver` running in the Internet Tablet. The application to be debugged is then ran in the Internet Tablet, which makes debugging results 100% accurate. This makes remote debugging the application using desktop PC possible, again the same connectivity as with the CPU transparency is needed when remotely debugging.

For more information about debugging, there is a comprehensive debugging guide on maemo.org website.

1.6 Application Packaging and Installing

Maemo uses the Debian package management system for installing and managing application packages and their dependencies. For end-user the actual package management is invisible and the application installation and removal in the Internet Tablet is done by Application Manager. The Debian package management system uses packages which consists of application binaries, optional libraries, meta data describing the package, dependencies to other packages and optional pre-install and post-install scripts. Packaging the applications is done with standard Debian packaging tools.

After creating the Debian package (creation is identical to the desktop Linux environment) the application is ready to be installed to Internet Tablet. Application is either copied to the device and installed using Application manager, or by placing the package into the `package repository` (essentially a web or FTP site containing application packages) and creating a `single-click install-file`. The single-click install-file eliminates the need for user to manually configure repositories into the Application manager, providing an easy-to-use way for end-user to install the application.

The site maemo.org/downloads contains a vast amount of applications, ready to be installed to Internet Tablet, and utilises the single-click install method.

The maemo Application Development course material contains more information how to create Debian installation packages for Internet Tablets.