maemo™

# maemo 4 Quick Start Guide

October 29, 2007

# Contents

# Chapter 1

# Introduction

This document strives to give an overall picture of the maemo platform to developers willing to bring their applications to the Nokia Internet Tablets. These mobile devices run a Linux-based operating system on ARM architecture. The maemo software development kit (SDK) is currently provided natively for desktop GNU/Linux distributions such as Debian and Ubuntu. The SDK can also be used in other operating systems through a virtual environment.

The main focus is to explain the basics needed to create and port maemo compatible software. It is simple to get started even if concepts like cross-compilation or touchscreen user interface design are not familiar to the reader. Those familiar with GNU/Linux, GTK+ toolkit and C programming language will feel right at home.

This guide is also a good starting point for software development for any platform or device related to the GNOME Mobile family. As well as giving an introduction on new frameworks, libraries and tools provided by maemo, overall design concepts for this new generation of internet-capable devices are explained.

## 1.1 History and Philosophy

The maemo platform was introduced when the first Nokia Internet Tablet was announced at the LinuxWorld Summit in New York City on May 25th, 2005. The design philosophy of maemo is to be a real desktop computer in the mobile hardware. The target is that the system is easily extendable, being easy to create new and bring existing desktop applications to the platform.

Maemo uses known open desktop frameworks to enable easy software portability and familiarity. This strategy is quite different compared to many Linux based media players and phones that are closed or require special development tools. About 90% of the maemo platform is open and the majority of that comes from upstream open source projects. The rest that is closed consists of e.g. parts of the user interface and device drivers, owned either by Nokia or third party providers.

One important philosophy is to enable easy development and hacking. Developers can make modifications to the platform, for instance introducing own kernel modules. Nokia also hosts an active open source maemo community around the platform (maemo.org).

## 1.2 Target audience

This guide should be interesting reading for any developer with a grasp on GNU/Linux and GNOME technologies. Thanks to the many links provided to external resources, it can even be considered a good starting point for total newcomers with a technical or developer interest.

The prerequisites for the application developer willing to make the most from this guide is basic GNU/Linux C programming knowledge. It is assumed that the most common tools such as *gcc* are known. Much of the maemo user interface development is *GTK+* based. If GTK+ development is not familiar to the reader, this document gives starting points to learn the framework. No previous maemo or embedded mobile device programming knowledge is required.

## 1.3 Reading instructions

The reader should first go through the following two chapters on terms and definitions and maemo architecture. After that the rest of the chapters can be read in the order reader is interested in.

### 1.3.1 For the impatient

Even though it is not recommended, it is possible to start hacking immediately with maemo SDK by following the instructions in the maemo tutorial[36].

# Chapter 2

# Terms and Definitions

- *ABI* - Application Binary Interface provides object code level compatibility.
- *API* - Application Programming Interface provides source code level compatibility.
- *applet* - A small application that integrates to *Hildon Desktop*.
- *ARMEL* - A name that e.g. Debian uses for the little endian ARM EABI (*ABI* for the ARM architecture).
- *devkit* - Part of the *maemo SDK* that contains software development tools. The SDK contains multiple devkits e.g. doctools devkit.
- *Hildon* - Application framework used in the *maemo platform*. Developed by Nokia and based on GNOME/GTK+ technologies, currently in the process of becoming an upstream project in gnome.org.
- *Hildon Desktop* - The main user interface component of the maemo release *Chinook*, rewrite of *maemo desktop*.
- *Internet Tablet* - Product category for Internet optimized mobile devices with touchscreen. The term was coined by Nokia but is being used more widely to include other devices.
- *initfs* - Initial file system used as the root file system during Linux kernel boot e.g. for hardware initialization (contains kernel modules and utilities for initializing them). Mounted after boot to /mnt/initfs.
- *maemo* - Software platform for mobile devices developed by Nokia, based on GNU/Linux and GNOME/GTK+ technologies. It includes proprietary components to make it work on the Nokia Internet Tablets.
- *maemo.org* - Developer community web site maintained by Nokia, main point of reference for open source and third party developers in general.
- *maemo desktop* - version of main user interface component of the maemo release *Bora*
- *maemo-af-desktop* - Same as *maemo desktop*.

- *maemo SDK* - Software Development Kit to create and port applications to the maemo platform using a PC.
- *Nokia Internet Tablet OS* - *maemo platform* + proprietary applications packaged to an official device image provided by Nokia.
- *OSSO* - Open Source Software Operations, Nokia organization developing and integrating software for Internet Tablets.
- *rootfs* - Root file system on the device.
- *rootstrap* - Part of the SDK that contains selected software components from rootfs. Rootstrap is the root file system of a target inside Scratchbox.
- *Sardine* - An experimental distribution based on Hildon for maemo, primarily of interest for developers who wish to test ”bleeding edge” features that are being developed for future releases of maemo.
- *toolchain* - Part of the SDK that contains ARM cross compilation tools like compiler and linker.

Maemo SDK releases

- *Mistral*: maemo 2.0 release. Corresponds to the Nokia Internet Tablet SE 2006 version 2.01.2006.26-8.
- *Scirocco*: maemo 2.1 release, including mainly bugfixes and some other enhancements. Corresponds to Nokia Internet Tablet SE 2006 version 2.2006.39-14.
- *Gregale*: maemo 2.2 release (bugfixes and enhancements)
- *Bora*: maemo 3.x releases. corresponds to Internet Tablet OS releases ”1.2006.47-20”, ”2.2006.51-6” (maemo 3.0), ”3.2007.10-7” (maemo 3.1) and ”4.2007.26-8”+”4.2007.38-2” (maemo 3.2)
- *Chinook*: maemo 4.0 release
- *Diablo*: future maemo release
- *Elephanta*: future maemo release

# Chapter 3

# Architecture

This chapter describes the high-level architecture of the maemo platform from an application developer's point of view. After that a comparison of maemo and popular Ubuntu and Debian desktop Linux distributions is given.

## 3.1 Kernel

Maemo uses a Linux 2.6 operating system kernel. Linux is an open source operating system developed by thousands of volunteers and companies that share their work under GNU GPL license. Architecturally Linux has a monolithic kernel. All kernel code is run under supervisor mode. Kernel can be extended at runtime by dynamically loadable kernel modules. Various APIs exist for device driver, file system and networking protocol modules. Developers can add new kernel modules.

The maemo kernel is based on the ARM kernel branch and can be modified, recompiled and flashed by a developer. The Kernel Guide for maemo[34] gives details of these procedures. Some of the modules like WLAN come as binary only which means that the module APIs should remain unchanged if kernel is changed by a developer.

## 3.2 Base distribution

Maemo is based to a large extent on the same open source components found in the Debian[7] Linux distribution. Maemo builds on GNU/Linux for operating system core and GNOME/GTK+ for user interface architecture.

Maemo uses the same component packaging system as Debian - *dpkg*-tool's binary packages. New packages can be installed, old ones can be removed and the whole system can be upgraded by the package management framework. The file system structure also comes from Debian.

In order to run the software on a Internet Tablet, various optimizations and enhancements have been made. These include power management related issues, touchscreen input, performance and size optimizations. In order to reduce space maemo uses a shell and command line tools from *Busybox*[4].

Also notably the maemo platform adheres closely to the GNOME Mobile Platform[16]. At the time of writing, maemo uses all the same components as this platform except for service discovery.

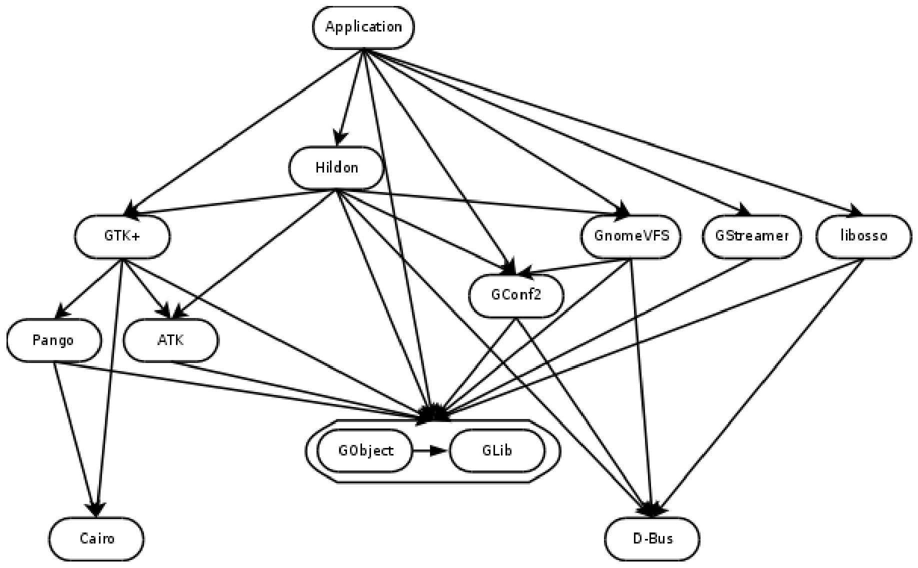
The maemo architecture guide[39] goes gives more detailed information of the overall maemo architecture.

## 3.3 Application framework

The purpose of an application framework is to help application development by providing a standard structure for an application. Applications that have a graphical user interface tend to have a similar structure, e.g. the event-driven runtime model. The events are triggered by the user, for example, by touching a button on the touchscreen. Events can also be triggered by the application engine itself. An example is when new data is received from the network. The application framework of maemo is called *Hildon*. It is partially based on the same technologies that *GNOME* framework is built on, most notably the *GTK+* components.

Hildon has several additions and enhancements to GNOME/GTK+ including Hildon widget set, *Sapwood* theme engine and image server, task navigator, Hildon control panel and status bar. Some of the changes to standard GNOME like Sapwood, for instance, are to reduce memory requirements and to improve speed on a small hand-held device. In addition, Hildon framework has many features to support mobility like automatic state saving, touchscreen input methods, and window management on a physically small device.

The programming APIs are familiar to GNOME and GTK+ developers. The framework has *GLib* and *GObject* object management system underneath. The GTK+ widget set is provided with Hildon extensions. The interprocess communication is done using *D-BUS* messages. The user files are accessed through *GNOME-VFS* and multimedia applications can use *GStreamer* to get accelerated support for various codecs. User configurations are stored via *GConf* and an XML parser API is available.



The figure above illustrates the most crucial components and their dependencies that the maemo application developer must deal with. These components are explained in more detail in the next chapters.

## 3.4 maemo compared to desktop Linux distributions

Ubuntu[71] is a popular Debian based Linux distribution for the ordinary desktop. The Ubuntu wiki[70] lists the key differences of Ubuntu and Debian. As one of the design principles of maemo has been to be as close as a traditional desktop Linux as feasible, here the differences between Debian/Ubuntu and maemo are explained in detail.

### 3.4.1 CPU architecture differences

Where as Debian supports multiple CPU architectures, Ubuntu's set is a bit more restricted. Compared to that maemo is an embedded ARM EABI[2] distribution. Maemo is also cross-compiled instead of natively compiled like Debian or Ubuntu. Maemo also uses different versions of toolchains (GCC, glibc[19] etc.) than Ubuntu or Debian for ARM feature support and maturity differences between architectures.

### 3.4.2 Security model differences

Instead of a multi-user system such as a traditional Linux desktop, maemo is considered a single user desktop system. The security model in maemo is focused on protecting the user from remote attacks and from his/herself, not from other users. Maemo also uses *suid* root binaries and */etc/password* where as Ubuntu enforces the use of *sudo* and *shadow passwords*.

Unlike Ubuntu, maemo makes use of a root account like Debian does but has a trivial default password. The user should really change the root password before installing e.g. *OpenSSH* to the device with root login.

### 3.4.3 Base system differences

The greatest difference in the base system is that maemo uses a lightweight BusyBox[4] replacement for the essential GNU utilities[20] on the device e.g. *ls* and *sh*. In maemo kernel and initfs reside in separate partitions and cannot be updated with a package manager like with a common desktop Linux. Programs in *initfs* use *uClibc*[72] instead of *glibc*. Ubuntu has Perl and Python languages as essential packages, Debian has only Perl and maemo has neither. Maemo has no *debconf*. Ubuntu uses *Upstart* for device startup instead of *SYSV init scripts* used in maemo.

# Chapter 4

# User Interface Development

The maemo platform's basic starting point for graphical user interface programming is the *Hildon*[27], an application framework comprising of a lightweight desktop, a set of widgets optimized for handheld devices, a set of theming tools and other complementary libraries and applications.

Hildon is based on GNOME[17] technologies to a great extent. Compared to a GNOME desktop from a user interface point of view, Hildon is designed to provide a new desktop for mobile embedded devices. Therefore it for example uses a lighter window manager called Matchbox[57].

## 4.1 Hildon desktop

The end-user's desktop experience on Nokia Internet Tablets is provided by the *Hildon Desktop* that is based on **Hildon**. The latter provides several libraries to interface with the desktop environment UI elements and services.

Hildon framework's main UI elements are separated to four categories and all of them can be extended with plug-ins:

- **Hildon Home** is the root desktop which can be customized by Hildon home applets.
- **Hildon Status bar** provides area for information and quick-access items used mainly to communicate device status changes.
- **Hildon Task Navigator** plugins implement the top level desktop menus.
- **Hildon Control Panel** is a general interface for application configuration and is extended by control panel plugins. See Hildon Desktop Plugins How-to[77] for more detailed information.

GUI applications in maemo usually have one or more *HildonWindow*s which are the top level application windows. They can include a standardized menubar and a toolbar.

The Hildon framework also includes other auxiliary widgets and services specialized for the Hildon environment. Examples of these are the *Hildon-FM* library for file system dialogs and widgets, *HildonBanner* for displaying notifications for user and *HildonWizardDialog* for creating wizard user interfaces. For information about Hildon's widgets good resources are Maemo Tutorial[36] and Maemo API References[24].

Another important element of the Hildon framework is the *Hildon Input Method* API, which is an interface for creating new user input systems in addition to the included virtual keyboard and handwriting recognition. Extending Hildon Input Methods How-to[11] describes the API's usage in detail.

## 4.2 Graphical user interface elements with GTK+

User interfaces of maemo applications are created using **GTK+**[26] user interface toolkit. GTK+ is widely used in UNIX GUI applications, perhaps most notably the GNOME desktop environment is based on it. The Hildon desktop builds on top of a modified GTK+ 2.10 called *maemo-GTK+ 2.10*[47] in a very similar manner - all Hildon widgets are also GTK+ widgets.

GTK+ itself is based on and uses several lower level libraries with all having a specific role. When developing applications using GTK+, usually the developer will also explicitly use some of these libraries.

- **GObject**[21] provides object orientation by *GType* system and signal handling.
- **GLib**[15] has tools for general non-graphical programming tasks such as different data structures, memory allocation, file operations, multithreading and interfacing with the underlying operating system.
- **XLib**[79] is a library for interacting with an X server
- **GdkPixPuf**[14] is used for simple client-side image manipulation.
- **GDK**[17] is a library between Xlib and GTK that is basically used for drawing widgets, handling window events and drag'n drop support
- **Pango**[62] is used for text rendering and layouting with solid internationalization support.
- **ATK**, Accessibility Toolkit is used for providing GTK+ widgets accessibility support.

More information about GTK+ and related libraries can be found from the project's web site[26] including GTK+ API Documentation[37].

# Chapter 5

# Important System Services

The underlying system services in the maemo platform differ slightly from those of used in desktop Linux distributions. This chapter gives an overview of the most important system services.

## 5.1 File system - GnomeVFS

Maemo includes a powerful file system framework GnomeVFS. This framework enables applications to use a vast number of different file access protocols without having to know anything about the underlying details. Some examples of the supported protocols are: local file system, HTTP, FTP and OBEX over Bluetooth.

In practice this means that all GnomeVFS file access methods are transparently available for both developer and end-user just by using the framework for file operations. The API for file handling is also much more flexible than the standard platform offerings. It features for example asynchronous reading and writing, MIME type support and file monitoring.

All user-file access should be done with GnomeVFS in maemo applications because file accesses can be remote. In fact many applications that come with the operating system on the internet tablets do make use of GnomeVFS. Access to files not visible to the user should be done directly for performance reasons.

A good hands-on starting point is taking a look of the GnomeVFS example in maemo-examples package. Detailed API information can be found from the GnomeVFS API reference[18].

## 5.2 Application Preferences - GConf

GConf is used by the GNOME desktop environment for storing shared configuration settings for the desktop and applications. The daemon process GConfd follows the changes in the database. When a change occurs in the database, the daemon applies the new settings to the applications using them. For example, the control panel application uses GConf.

If setting are used only by a single application, Glib utility for .ini style files should be used instead. Applications should naturally have working default settings. Settings should be saved only when the user changes them.

## 5.3 D-Bus

For interprocess communications (IPC), maemo relies heavily on D-Bus. D-Bus makes it possible for programs to export their programming interfaces so that other processes can call them in consistent manner without having to define a custom IPC protocol. Using these exported APIs is also language agnostic, so as long as programming language supports D-Bus, it can also access the interfaces.

A maemo specific library called *libosso* provides helpful wrappers for D-BUS communication. It also contains required functionality for every maemo application. Applications must be initialized using this library. With it applications can connect to listen to system hardware state messages such as ”battery low”. The library is used also for application state saving and auto-save functionality. The maemo tutorial[36] provides a good introduction to libosso.

## 5.4 Other system services

Maemo platform provides many libraries and APIs for different common tasks. Alarm framework and GPS API are described here, other relevant services are described later in this document.

### 5.4.1 Alarm Framework

The Alarm framework included in maemo gives developers an easy to use an interface for working with all kinds of timed events. It provides a consistent user interface, audio playing, starting the device, sending a D-BUS signal or executing a file and more. The document Using Alarm Interface[73] describes the framework in detail.

### 5.4.2 GPS API

The GPS framework in maemo consists of a GPS daemon and a library for controlling it. The maemo connectivity guide[41] gives examples how to use the library in applications.

# Chapter 6

# Multimedia

Maemo offers lots of possibilities for multimedia applications. Maemo was built with the Internet Tablet devices in mind - hardware in this form factor provides a big high-resolution touch screen, audio input and output, video and image capturing through the camera, fast network connections for streaming and a digital signal processor for efficient audio and video manipulation.

For developers there are open programming interfaces to make use of these features apart from directly programming the DSP. Preferred way of doing video and audio programming for maemo is using the GStreamer framework and for manipulating static images there are several libraries. Interactive multimedia for applications like games can be done using SDL framework. The Multimedia Architecture document[59] gives more details on how the components are organized both on the physical device and on the maemo SDK.

## 6.1 GStreamer - multimedia framework

The main multimedia framework in maemo is GStreamer. It is based on the concept of pipelines which are made of multiple elements. Elements themselves can be practically anything that do something with a data stream.

Maemo includes a variety of these elements to support audio and video effects, encoding and decoding and interfacing with the device's hardware. Using this approach, developer does not have to bother with details like low level hardware management, audio and video compression schemes etc. If the elements included with the OS are not enough, more can be either compiled for maemo or new ones can be developed.

Developer documentation for GStreamer can be found at the project's website[23].

## 6.2 Stream encoding and decoding

Maemo devices include a wide arsenal of video and audio codecs which enables the maemo applications to read and write nearly all commonly used video and audio formats. These are supported also by the GStreamer framework. For many purposes the developer doesn't even have to specify the used codecs implicitly since GStreamer automatically detects the format and which codec to use. The *playbin*[22] GStreamer base plugin provides a convenient abstraction layer for all audio and video content.

Due to non-technical reasons, most of the codecs are not distributed with the SDK which is good to keep in mind when developing applications relying on such features.

## 6.3 Audio

For audio programming maemo has two main APIs: GStreamer and ESound. Usually system sounds, such as sounds for notifying user for an event, e.g. battery low, is played through ESound. More sophisticated operations, for example playing music files or recording audio, should be generally done using GStreamer which provides better performance and a much more flexible API. Most of the maemo's computing intensive GStreamer elements are implemented using the device's DSP which greatly enhances their performance.

Linux kernel also has two lower level audio interfaces: ALSA and OSS. From these ALSA is supported through a plug-in package which is part of the SDK. The legacy API OSS is not supported by the kernel, but ALSA has an OSS emulation system, which works for most purposes.

For the audio APIs' documentation, see the GStreamer web site[23], ESound white paper[9] and ALSA project's web site[1]. More in-depth material is available in the Multimedia Architecture guide[59].

## 6.4 Video

Although the framework hides much of the implementation and hardware details, it's good for a developer to know what happens beneath the interfaces. Video capturing is done via a Linux kernel's Video4Linux API and graphics are displayed using the X Window System. Practically all GNU/Linux applications rely on these components for video tasks so porting existing applications should be quite effortless and finding support easy.

Hands on instructions for using capture and output features are given in the Maemo Camera How-to[30].

## 6.5 Digital Signal Processor

Under the hood Internet Tablets have a dedicated digital signal processor (DSP). It's design is optimized for tasks like stream coding and audio effects. Maemo has a high level programming support for the DSP in form of GStreamer elements which can decode most of the supported file formats. By using the DSP computing load is also taken off from the main processor which greatly enhances system performance and responsivity.

## 6.6 Graphics and Images

Maemo includes several libraries for working with images and graphics:

- Cairo is a library for doing 2D vector graphics. It features for example high quality vector based graphics, support for importing and exporting variety of formats such as SVG, PNG, PDF and Postscript and bindings for many programming languages

- GDK and GdkPixbuf are the bitmap graphics libraries that GTK+ is built on. They together provide opening and saving bitmap images in JPEG, PNG, TIFF, ICO and BMP formats, tight integration with GTK+ and tools for drawing with graphic primitives. Also e.g. loading of SVG images is supported
- For more detailed control over the different image formats, there are many more specialized libraries, for example libpng, libjpeg and libwmf.

Visit Cairo website[5] for guides, tutorials and references. GDK's and GdkPixbuf's documentation can be found at GTK+-project's website[25].

## 6.7 Games

For game developers maemo offers a common startup and settings screen and a framework called *osso-games-startup*. Using this library game developers and porters can reduce the common startup screen coding and concentrate on the real game programming. osso-games-startup's operating system communication features also offer an interface to make games behave correctly. For example it eases the management of full screen mode and exceptional situations, such as battery low notification, in a unified and user friendly manner. See Using Games Start-up Screen document[74] for more in-depth description and API usage.

# Chapter 7

# Communications

For communications with the outside world maemo platform provides frameworks ranging from Bluetooth file transfer to making video calls. This chapter goes through the components important for the maemo application developer.

## 7.1 Acquiring internet access

*LibConIC* is the library that all applications that want to use internet connectivity must use. It takes care of e.g. scanning of available WLAN networks and setting up the IP network after user has selected a connection. LibConIC API works together with the maemo connectivity daemon (*ICd*) that handles both WLAN and Bluetooth connections. LibConIC and ICd are described more deeply in the maemo connectivity guide[41].

## 7.2 VoIP, Instant Messaging and Presence

The *Telepathy*[69] communications framework provides an unified API for presence, messaging and voice/video calls. The list of supported protocols is long: IRC, ICQ, XMPP (Jabber), SIP, MSN etc. The connection manager of Telepathy is expandable and uses D-BUS for communication. Telepathy usage in maemo is described more deeply in in ”implementing custom connection managers”[32].

## 7.3 Address Book and Contacts

The maemo platform features a centralized storage for address book data which is implemented using the Evolution Data Server (EDS). This allows different applications to effortlessly share contacts in a standardized form so that the user doesn't have to specify them in every program separately.

For accessing an EDS contact from an application data maemo has the *libosso-abook* library which has powerful widgets that enable consistent user interface for accessing and managing the contact information.

Good reading to quickly start using libosso-abook is the Using Maemo Address Book APIs document[75]. More in depth information can be found at the Maemo API

Reference[41] and the Evolution project's web site[10]. *maemo-examples* package also includes commented example code that covers using the address book APIs.

## 7.4 Bluetooth

A high level API for Bluetooth is offered as part of the maemo connectivity subsystem. Using its D-BUS API a program can find remote Bluetooth devices such as phones, send files over OBEX object push and create pairings with remote devices. For these tasks it's recommended for application to use this framework as it not only has a lot simpler API but makes the applications look and behave consistently.

For Bluetooth operations that aren't supported by the maemo connectivity framework maemo includes a lower level BlueZ D-BUS API, which is also the main Bluetooth interface for all Linux systems. The BlueZ API has features for practically all aspects of Bluetooth systems, and as a consequence its a lot more complex than the higher level Maemo Connectivity subsystem's offerings.

The Maemo Connectivity Guide[41] describes the high level D-BUS API and its usage. More information about the BlueZ API can be found at BlueZ web site[3]. The maemo-example package also includes example code about both libraries.

### 7.4.1 OBEX

Bluetooth devices use *OBEX* protocol to exchange data objects. Maemo includes libraries to help working with this protocol. For OBEX FTP the easiest way is to use GnomeVFS's OBEX backend. For more granular control there is *libgwobex* which GnomeVFS uses in its implementation and even more low level interface is the OpenOBEX library.

More about GnomeVFS is written in this document's GnomeVFS section. Maemo Connectivity Guide[41] has chapters for libgwobex and *OpenOBEX*. For libgwobex API see Maemo API Reference[37] and lots of information about OpenOBEX can be found at the project's web site[60].

# Chapter 8

# Development Environment

The development environment for maemo running on the desktop is called *maemo SDK*[54]. It will only install and run on a Linux operating system. Supported Linux distributions for maemo SDK are currently Debian and Ubuntu, but installing maemo SDK is also possible for other distributions. On other operating systems such as Windows, a VMWare image[80] can be used to provide a working Linux environment.

## 8.1 maemo software development kit

The maemo SDK creates a sandboxed maemo development environment on a GNU/Linux desktop system largely built on a tool called *Scratchbox*[66]. In most ways this environment behaves like the operating system on the device but with added development tools. This means that the development process is very similar to normal desktop GNU/Linux and the kinks of embedded development, such as cross-compiling, are handled transparently by Scratchbox.

### 8.1.1 Hardware architectures

Maemo SDK comes with two environments for two architectures. *x86* is used for native performance and better tool support though native execution without need for emulation. *ARMEL* is used for working with the actual device's architecture. Both have their advantages and roles in maemo development. It is important to understand that maemo SDK actually provides these two environments as preconfigured *targets* inside a working Scratchbox installation. This is explained in detail later.

Generally in active development the x86 environment is used because it provides practically the same performance as normal GNU/Linux applications. Also, although the underlying architecture is different from the actual device, programs usually behave exactly as they would when compiled and run on ARMEL. As stated before, many tools are available only for the x86 environment.

When an application is running fine in the x86 environment the next step is to compile it for the ARMEL architecture. The process for compilation and packaging is exactly the same as in x86, albeit a bit slower because some of the required software is emulated. The developer needs not to concern with cross-compilation. This is the main rationale for maemo SDK to use Scratchbox in the first place.

The applications compiled in ARMEL environment can be run straight on the device. Additionally, it is possible to run some of the applications inside the ARMEL environment in maemo SDK. This is possible because of the automatic emulation maemo SDK provides. The emulation is not complete and things like multithreading will cause problems, so actual testing must be done on the device.

Using emulation for the whole development process may not sound ideal because of the effect on performance. That's why even when in ARMEL environment, native performance is achieved with Scratchbox by internally using the host computer's tools without emulation when possible. For example, compilation on ARMEL environment is actually done by an x86-ARMEL cross-compiler, but Scratchbox hides the details so that the developer can execute GCC like on any GNU/Linux system.

### 8.1.2 Development on maemo SDK

The maemo SDK provides all of its development tools inside Scratchbox. Also the Hildon Desktop is started with a single command *af-sb-init start* inside Scratchbox. However, the Hildon Desktop needs a secondary *X server* of proper size and bitdepth to be displayed on.

As an exception to the rule, the X server such as *Xephyr*[78] must be started on the host Linux environment instead inside Scratchbox. The use of Xephyr is described in the maemo SDK tutorial[36].

### 8.1.3 Development tools on Scratchbox

As the Scratchbox environment is practically a full GNU/Linux system, it includes the standard GNU/Linux development tools. Debugging is done with tools like *GDB*[13], *valgrind*[76], *ltrace* and *strace*[68]. Performance profiling can be done with tools like *htop*[31], *oprofile*[61] and *time* and compiling with the GCC toolchain. Some of these tools offer graphical user interfaces which can also be used. Of course this isn't a comprehensive list of the tools and if the tools shipped with the SDK don't suit to needs or personal preferences, most utilities can be easily run practically unchanged in Scratchbox.

The Maemo Tutorial[36] is a good introduction to development using Scratchbox and debugging tools are described in depth in The Maemo Debugging Guide[42].

### 8.1.4 Testing and debugging on device

Even though Scratchbox is quite accurate in emulating a full target environment on the device, it isn't 100% identical. Especially applications that make use of the device's special hardware can behave differently on the device than on Scratchbox. They even may not work at all. Fortunately testing the software on device is quite straightforward using either SSH or a tool called *sbrsh* to run target binaries on the device transparently from Scratchbox.

The CPU Transparency how-to[6] has instructions for getting started with sbrsh. SSH server and client for maemo can be downloaded from maemo web site[48].

### 8.1.5 Scratchbox under the hood

Scratchbox is maemo SDK's cross-compiling environment. The default Scratchbox installation works as-is under most conditions, but some details are good to know for more specialized usage.

The *target* inside Scratchbox contains a root file system that is being worked on. When a new target inside Scratchbox is created, a *toolchain* must be specified for it. Using this toolchain, applications are built for the target. Examples of a *target* are X86 and ARMEL which are provided by the maemo SDK on top of Scratchbox.

*Host tools* are native to the host provided for convenience and speed. They are always preferred over target tools and transparently for example cross-compile applications to the target architecture. Host tools consist of devkits and *toolchain*s.

A toolchain provides the minimal set of tools for compiling binaries for the target. One and only one toolchain must be selected for every Scratchbox target.

*CPU transparency methods* take care of running the applications on an emulator, target device or directly on the host transparently to the user. The available CPU transparency methods come from a special *devkit* called *cputransp*. For each of maemo SDK's predefined targets a CPU transparency method is selected and defined.

A *toolchain* is a collection of tools used to produce binaries for the target environment. In addition to a compiler (*gcc*) it contains a linker (*ld*) and other *binutils* such as *strip*, *objdump* and *strings*.

A *devkit* is a collection of tools native to the host. A toolkit can be selected or disabled for a target. An example of a devkit is doctools devkit, which provides tools (like doxygen) for building documentation.

A *rootstrap* is a root file system for the target device. Maemo SDK provides root file systems for both targets (X86 and ARMEL) inside Scratchbox. Note that the user's home directory is shared for all targets. The */tmp* directory is shared for all targets and also with host.

From Scratchbox's point of view, Maemo SDK is a set of preconfigured *target*s and *root file system*s. One set is provided for both *X86* and *ARMEL* architectures on top of a working Scratchbox installation.

More Scratchbox information can be found at Scratchbox's web site[66].

## 8.2 Packaging, deploying and distributing

For installing and managing application packages maemo uses the popular Debian package management system. From the user's perspective this is quite invisible as the package management is done using the Application Manager. Under the hood it presents a flexible package framework which enables developers to easily create installable and manageable packages without having to concentrate on the actual implementation details of the management.

### 8.2.1 Deb installation packages

The Debian package management system uses deb-packages which, in addition to files that will installed, consist of meta data describing the package, dependencies to other packages and optional installation and removal scripts. The process of creating the packages is fortunately made quite simple by various package development tools.

The process of creating packages is described in the Making Application Packages document[56].

### 8.2.2 Package repositories

The package management system makes use of package repositories, which are essentially web or FTP sites that contain packages and some information about them. It isn't mandatory for a package to exist in any repository, but it has significant advantages: user can easily update packages, other developers can use the packages as automatically installable dependencies and the repository's packages can be found using Application Manager.

Information for working with repositories can be found in the Debian Repository HOWTO[8].

### 8.2.3 One-click-install

Maemo also has a option to create installation instruction files that enables the users to install applications simply by clicking a link on a web site. The .install files contain required repository and package names. The format of these files is really simple and it's described in Hildon Application Manager documentation[28].

## 8.3 Programming languages

Currently C is the only official programming language for maemo. But thanks to the community, unofficial support exist for several other languages. To name a few, the SDK itself compiles C++ and by adding hildonmm bindings[29] Hildon applications can be created the C++ way. Python scripting language also has a good support in form of pymaemo[63] and Ruby bindings are in the works[65], not to forget Mono[58]. For Java support, JaliMo[33] is an interesting project to track.

# Chapter 9

# Porting Software

Much effort has been done in the design of maemo platform to allow easy porting of regular GNU/Linux desktop software to the mobile maemo environment. Earlier in this guide the basic tools that ease cross-compilation and help coping with the GNU autotools were explained. This chapter gives pointers to the relevant guides focusing on the differences in the application programming and user interfaces.

## 9.1 Command line programs

Porting software with no user interface in most cases is trivial and straight forward. First, the source code of a program is unpacked to the home directory of a Scratchbox user. Second, inside ARMEL target *configure* and *make* are run. Then the compiled program can be tested on the device. Finally the software needs to be packaged.

## 9.2 Programs with a graphical user interface

The porting of an application that uses GTK+ for its graphical user interface begins the same way as above. In addition to this, the user interface part needs to be refactored to use Hildon instead of directly using GTK+. Dependencies to GNOME components, if any, need to be removed or replaced with corresponding maemo SDK components. If the application uses any components not available in the maemo SDK, these must be also ported by the developer.

Applications that do not use GTK+ and instead use e.g. SDL need more work. The first thing that is easily noticed that the virtual keyboard is missing because the Hildon Input Method is not available. A user of the application cannot interact with the application.

The maemo 4.0 porting guide[35] goes into the necessary details of porting an existing GTK+ application to maemo environment.

## 9.3 Localization

The localization of maemo applications is done using the common *gettext* package. The translating of an application to different languages is described in detail in the Localization How-to[50].

# Chapter 10

# Quality Considerations

The maemo developer needs to take care many details when developing a good quality application for an embedded device. Compared to the desktop Linux, a developer needs to be more careful with performance, resource usage, power consumption, security and usability issues. The Quality Awareness project[64] provides a generic test plan for developers to help them to create a good quality software.

## 10.1 Performance and Responsiveness

The user wants to be in control: the device must be responsive at all times and give appropriate feedback. The UI responsiveness can be achieved e.g. in the three following ways: First, using process events by showing a banner and blocking the UI while operating. Second option would be to slice the operation into parts while updating e.g. a progress bar. Thirdly, threading can be achieved with the help of GLib but this is increasingly hard to implement and debug. For time consuming tasks the second option is recommended. If the operation takes only a second or two, the first option can also be used.

As other general guidelines, CPU over usage must naturally be avoided. This goes also for completely redundant operations such as unnecessary screen updates. Also, when an application is not visible, it should be inactive to prevent it from slowing down the application user is currently interacting with.

## 10.2 Resource Usage

Resource usage has to be taken into account with maemo applications very carefully, because if an application takes up too much resources, it cannot be used. The resources that are scarce are RAM, flash, network bandwidth, file descriptors and IO bandwidth. Also besides base memory usage also memory leaks must be naturally avoided. There are tools such as Valgrind[76] to help detecting possible memory leaks.

## 10.3 Power consumption

The device is not usable if user needs to recharge it all the time or if it runs out of battery while idle. Therefore, no polling or busy looping should be used anywhere.

Instead, an event based approach like GTK's main loop is a better alternative. Also, too frequent timers or timeouts are to be avoided.

## 10.4 Secure Software Design

Quality and robustness together produce secure software. A document[67] describing secure software design on maemo platform will give details about creating secure software and the specific issues on mobile platforms.

# Chapter 11

# maemo.org

The landing site and home for maemo open source community developers is maemo.org. It provides documentation, community support, bug reporting and complete project hosting for the developers. This chapter goes though these services.

## 11.1 Documentation

The maemo.org site provides official documentation[43] for each maemo major release in the form of tutorials, how-tos and API documentation. Also this Quick Start Guide is part of the official documentation.

The community documentation is viewable for everyone and editable by registered users in the maemo wiki[55]. Contributions to the community wiki and feedback about the official document are welcome!

## 11.2 Support

The place to get support from the community is the mailing list for maemo developers[51]. Prior to asking a possibly frequently asked question on the list, it is good to take a look at the mailing list archives[44]. A separate list for maemo users is also very active targeted more to the end users than maemo application developers.

The most active maemo instant messaging channel at the time of writing is the maemo *IRC channel*[49] on *freenode*.

Commercial support for the maemo platform will be provided by *Forum Nokia*[12].

## 11.3 Bug Reporting

The proper way of giving feedback about the maemo platform is by using the public maemo Bugzilla[40]. It can be browsed anonymously, but registration is needed to be able to enter new bug reports or feature requests. The same issue management system is used to give feedback about the software running in the device, the SDK and the website including its documentation.

## 11.4 Project Hosting

The *maemo garage*[46] is a complete project hosting site for maemo community. This is the place where the actual development of maemo applications is preferably done, hence the name. Garage provides the developers for each project a version control management system, wiki, issue tracker and forums.

*Application catalog*[45] or the *Downloads* section of the maemo site is meant for end users to be able to easily download and install open source software products on their Internet Tablets. The maemo Application Catalog manual[38] has instructions for the maintainer on how to add a new tested application to the application catalog list, hopefully incubated first in the garage.

## 11.5 Roadmap and News

The maemo community members can keep a track on what is happening now by reading the news[52]. For future plans, the roadmap[53] is kept open for avoiding overlapping work and giving visibility where the development effort is going for maemo itself.

# Bibliography

[1] Alsa project's home page. http://www.alsa-project.org/.

[2] Application binary interface (abi) for the arm architecture. http://www.arm.com/products/DevTools/ABI.html.

[3] Bluez project's home page. http://www.bluez.org/.

[4] Busybox. http://www.busybox.net/.

[5] Cairo project's home page. http://cairographics.org/.

[6] Cpu transparency how-to. http://maemo.org/development/documentation/how-tos/4-x/cpu_transparency_how-to.html.

[7] Debian - the universal operating system. http://www.debian.org/.

[8] Debian repository howto. http://www.debian.org/doc/manuals/repository-howto/repository-howto.

[9] Esound white paper. http://developer.gnome.org/doc/whitepapers/esd/.

[10] Evolution project's home page. http://www.gnome.org/projects/evolution/.

[11] Extending hildon input methods. http://maemo.org/development/documentation/how-tos/4-x/extending_hildon_input_methods.html.

[12] Forum nokia. http://www.forum.nokia.com/.

[13] Gdb: The gnu project debugger. http://sourceware.org/gdb/.

[14] Gdkpixpuf library. http://library.gnome.org/devel/gdk-pixbuf/.

[15] Glib reference manual. http://developer.gnome.org/doc/API/glib/.

[16] Gnome mobile. http://www.gnome.org/mobile/.

[17] Gnome: The free software desktop project. http://www.gnome.org/.

[18] Gnomevfs api reference. http://developer.gnome.org/doc/API/gnome-vfs/.

[19] Gnu c library. http://www.gnu.org/software/libc/.

[20] Gnu core utilities. http://www.gnu.org/software/coreutils/.

[21] Gobject reference manual. http://library.gnome.org/devel/gobject/.

[22] Gstreamer playbin base plugin. http://gstreamer.freedesktop.org/data/doc/gstreamer/head/gst-plugins-base-plugins/html/gst-plugins-base-plugins-playbin.html.

[23] Gstreamer project's home page. http://gstreamer.freedesktop.org/.

[24] Gtk+ api documentation. http://www.gtk.org/api/.

[25] Gtk+ project's home page. http://www.gtk.org/.

[26] Gtk+ web site. http://www.gtk.org/.

[27] Hildon. http://live.gnome.org/Hildon/.

[28] Hildon application manager documentation. http://hildon-app-mgr.garage.maemo.org/doc.html.

[29] Hildonmm project page. https://garage.maemo.org/projects/maemomm/.

[30] How to use camera api. http://maemo.org/development/documentation/how-tos/4-x/how_to_use_camera_api.html.

[31] htop - an interactive process viewer for linux. http://htop.sourceforge.net/.

[32] Implementing custom connection managers. http://maemo.org/development/documentation/how-tos/4-x/implementing_custom_connection_managers.html.

[33] Jalimo. http://www.jalimo.org/.

[34] Kernel guide for maemo. http://maemo.org/development/documentation/how-tos/4-x/kernel_guide_for_maemo.html.

[35] maemo 4.0 porting guide. http://maemo.org/development/documentation/how-tos/4-x/maemo_4-0_porting_guide.html.

[36] Maemo 4.0 tutorial. http://maemo.org/development/documentation/tutorials/maemo_4-0_tutorial.html.

[37] Maemo api reference. http://maemo.org/development/documentation/apis/.

[38] maemo application catalog manual. http://maemo.org/community/application-catalog/user-manual.html.

[39] maemo architecture. http://maemo.org/development/documentation/how-tos/4-x/maemo_architecture.html.

[40] maemo bugzilla. http://bugs.maemo.org.

[41] Maemo connectivity guide. http://maemo.org/development/documentation/how-tos/4-x/maemo_connectivity_guide.html.

[42] Maemo debugging guide. http://maemo.org/development/documentation/how-tos/4-x/maemo_debugging_guide.html.

[43] maemo developer documentation. http://maemo.org/development/documentation/.

[44] maemo-developers mailing list archive. http://lists.maemo.org/pipermail//maemo-developers/.

[45] maemo downloads. http://maemo.org/downloads/.

[46] maemo garage. http://garage.maemo.org.

[47] maemo-gtk+ 2.10. http://live.gnome.org/Maemo/Gtk210Changes.

[48] maemo home page. http://maemo.org/.

[49] maemo irc channel. http://maemo.org/community/irc.html.

[50] maemo localization how-to. http://maemo.org/development/documentation/how-tos/4-x/maemo_localization_how-to.html.

[51] maemo mailing lists. http://maemo.org/community/mailing-lists.html.

[52] maemo news. http://maemo.org/news/.

[53] maemo roadmap. http://maemo.org/intro/roadmap.html.

[54] maemo sdk releases. http://maemo.org/development/sdks/.

[55] maemo wiki. http://maemo.org/community/wiki.

[56] Making application packages. http://maemo.org/development/documentation/how-tos/4-x/making_application_packages.html.

[57] Matchbox home page. http://matchbox-project.org/.

[58] Mono. http://www.mono-project.com/Maemo.

[59] Multimedia architecture. http://maemo.org/development/documentation/how-tos/4-x/multimedia_architecture.html.

[60] Openobex home page. http://dev.zuckschwerdt.org/openobex/.

[61] Oprofile - a system profiler for linux. http://oprofile.sourceforge.net/.

[62] Pango reference manual. http://library.gnome.org/devel/pango/.

[63] Pymaemo home page. http://pymaemo.garage.maemo.org/.

[64] Quality awareness. http://maemo.org/development/documentation/how-tos/4-x/quality_awareness.html.

[65] Ruby maemo bindings project page. https://garage.maemo.org/projects/ruby185/.

[66] Scratchbox home page. http://scratchbox.org/.

[67] Secure software design. TODOnotpublishedwithmaemoChinookBETA.

[68] strace. http://sourceforge.net/projects/strace/.

[69] Telepathy. http://telepathy.freedesktop.org/.

[70] Ubuntu for debian developers. https://wiki.ubuntu.com/UbuntuForDebianDevelopers.

[71] Ubuntu home page. http://www.ubuntu.com/.

[72] uclibc. http://www.uclibc.org/.

[73] Using alarm interface. http://maemo.org/development/documentation/how-tos/4-x/using_alarm_interface.html.

[74] Using games start-up screen. http://maemo.org/development/documentation/how-tos/4-x/using_games_start-up_screen.html.

[75] Using maemo address book apis. http://maemo.org/development/documentation/how-tos/4-x/using_maemo_address_book_apis.html.

[76] Valgrind. http://valgrind.org/.

[77] Writing hildon desktop plug-ins for maemo. http://maemo.org/development/documentation/how-tos/4-x/writing_hildon%_desktop_plug-ins_for_maemo.html.

[78] Xephyr. http://www.freedesktop.org/wiki/Software/Xephyr.

[79] Xlib manual. http://www.the-labs.com/X11/XLib-Manual/.

[80] Xubuntu image with maemo sdk installed. https://garage.maemo.org/frs/?group_id=277.