GTK+ / Gnome Application Development

Havoc Pennington
Red Hat Advanced Development Labs

GTK+ / Gnome Application Development
by Havoc Pennington

First Release Edition
Published 1999

Table of Contents

I. Overview

Table of Contents

The first chapter of the book gives you an overview of GTK+ and Gnome technologies. Chapters 2 and 3 rapidly cover the basics of GTK+ programming. These chapters move very quickly, and give you the information you need to understand the remainder of the book. For a gentler introduction to GTK+, you may wish to read the GTK+ Tutorial or Eric Harlow's book Developing Linux Applications with GTK+ and GDK.

Chapter 1. Introduction

This chapter gives you an overview of the technologies described in this book.

1.1. What is Gnome?

Gnome is a free (or "open source") software development project started in 1997 by Miguel de Icaza of the Mexican Autonomous National University and a small team of programmers from around the world. Inspired by the success of the similar K Desktop Environment (KDE) project, the burgeoning popularity of the GNU/Linux operating system, and and the power of the GTK+ graphical toolkit, Gnome grew quickly within a year, hundreds of programmers were involved and many thousands of lines of code had been written. Gnome has become a powerful framework for GUI application development which runs on any modern variety of UNIX.

"Gnome" is actually an acronym: GNU Network Object Model Environment. Originally, the project was intended to create a framework for application objects, similar to Microsoft's OLE and COM technologies. However, the scope of the project rapidly expanded; it became clear that substantial groundwork was required before the "network object" part of the name could become reality. The latest development versions of Gnome include an object embedding architecture called Bonobo, and Gnome 1.0 included a fast, light CORBA 2.2 ORB called ORBit.

Gnome is a part of the GNU Project, whose overall goal is developing a free operating system (named GNU) plus applications to go with it. GNU stands for "GNU's Not UNIX", a humorous way of saying that the GNU operating system is UNIX-compatible. You can learn more about GNU at http://www.gnu.org.

Gnome has two important faces. From the user's perspective, it is an integrated desktop environment and application suite. From the programmer's perspective, it is an application development framework (made up of numerous useful libraries). Applications written with the Gnome libraries run fine even if the user isn't running the desktop environment, but they integrate nicely with the Gnome desktop if it's available.

The desktop environment includes a file manager, a "panel" for task switching, launching programs, and docking applets, a "control center" for configuration, and several smaller bells and whistles. These programs hide the traditional UNIX shell behind an easy-to-use graphical interface.

Gnome's development framework makes it possible to write consistent, easy-to-use, interoperable applications. The X Window System designers made a deliberate decision not to impose any user interface policy on developers; Gnome adds a "policy layer," creating a consistent look-and-feel. Finished Gnome applications work well with the Gnome desktop, but can also be used "standalone" users only need to install Gnome's shared libraries. It's even possible to write Gnome applications which do not rely on the X Window System; you might want to provide a non-graphical CORBA service, for example.

This book is about Gnome from a developer's point of view; it describes how to write a Gnome application using the Gnome libraries and tools.

1.2. The Gnome Development Framework

Gnome's application development framework centers around a suite of libraries, all written in portable ANSI C and intended to be used on UNIX-like systems. Libraries which involve graphics realy on the X Window System. Wrappers are available which export the Gnome API to nearly any language you can think of, including Ada, Scheme, Python, Perl, Tom, Eiffel, Dylan, and Objective C. There are at least three different C++ wrappers as well.

This book will cover the C interface to the libraries; however, it should be useful for users of any language binding, since the mapping from C to your preferred language is typically straightforward. The book covers version 1.0 of the Gnome libraries (including the compatible bug fix releases, such as 1.0.9all 1.0.x versions are compatible).

1.2.1. Non-Gnome Libraries

Taking full advantage of the free software tradition, Gnome didn't start from scratch. It uses several libraries which are maintained separately from the Gnome project. These are a part of the Gnome application development framework, and you can count on their presence in a Gnome environment.

1.2.1.1. glib

glib is the base of the Gnome infrastructure. It's a C utility library, providing routines to create and manipulate common data structures. It also addresses portability issues; for example, many systems lack the snprintf() function, but glib contains an implementation called g_snprintf() which is both guaranteed to exist on all platforms and slightly safer than snprintf() (it always NULL-terminates the target buffer).

Gnome 1.0 uses glib version 1.2 and works with any glib in the 1.2 series (1.2.1, 1.2.2, etc.). All glib versions beginning with 1.2 are compatible bug-fix releases.

1.2.1.2. GTK+

GTK+, or the Gimp Tool Kit, is the GUI toolkit used in Gnome applications. GTK+ was originally written for the Gimp (GNU Image Manipulation Program http://www.gimp.org), but has become a general-purpose library. GTK+ depends on glib.

The GTK+ package includes GDK, the Gimp Drawing Kit, which is a simplification and abstraction of the low-level X Window System libraries. Since GTK+ uses GDK rather than calling X directly, a port of GDK permits GTK+ to run on windowing systems other than X with relatively few modifications. GTK+ and the Gimp have already been ported to the Win32 platform in this way.

GTK+ provides several features for Gnome applications:

- A dynamic type system.

- An object system written in C, complete with inheritance, type checking, and a signal/callback infrastructure. The type and object systems are not GUI-specific.

- A GtkWidget object written using the object system, which defines the interface GTK+'s graphical components implement.

- A large collection of useful GtkWidget subclasses (widgets); this collection forms the bulk of GTK+'s code.

Gnome adds a number of additional widgets to the basic GTK+ collection.

Gnome 1.0 is based on GTK+ version 1.2. All GTK+ versions beginning with 1.2 are compatible bug-fix releases; 1.2.1, for example.

1.2.1.3. ORBit

ORBit is a CORBA 2.2 ORB written in C. It was designed to be small and fast compared to other ORBs, and supports the C language mapping. ORBit is implemented as a suite of libraries.

CORBA, or Common Object Request Broker Architecture, is a specification for Object Request Brokers, or ORBs. An ORB is much like a dynamic linker, but it works with objects, rather than subroutines. At runtime, a program can request the services of a particular object; the ORB locates the object and creates a connection between it and the program. For example, an email program might request an "addressbook" object, and use it to look up a person's name. Unlike dynamic linking, CORBA works fine across a network, and even allows different programming languages and operating systems to interact with one another. If you're familiar with DCOM on the Windows operating system, CORBA is analagous.

1.2.1.4. Imlib

Imlib ("Image Library") provides routines for loading, saving, displaying, and scaling images in a variety of popular formats (including GIF, JPEG, PNG, and TIFF). It comes in two versions; an Xlib-only version, and a GDK-based version. Gnome uses the GDK version.

1.2.2. Gnome Libraries

The libraries in this section are a part of the gnome-libs package and were developed specifically for the Gnome Project.

1.2.2.1. libgnome

libgnome is a collection of non-GUI-related routines for use by Gnome applications. It includes code to parse configuration files, for example. It also includes interfaces to some external facilities, such as internationalization (via the GNU gettext package), argument parsing (via the popt package), and sound (via the Enlightenment Sound Daemon, esound). The gnome-libs package takes care of interacting with these external libraries, so the programmer does not need to concern herself with their implementation or availability.

1.2.2.2. libgnomeui

libgnomeui collects GUI-related Gnome code. It consists primarily of widgets designed to enhance and extend GTK+. Gnome widgets generally impose user interface policy, which permits a more convenient API (since there is less for the programmer to specify). It also results in applications with more consistent interfaces, of course.

Highlights of libgnomeui include:

- The GnomeApp widget, which makes it easy to create a nice main window for your application. It uses another widget called GnomeDock which enables users to rearrange and "undock" toolbars.
- The GnomeCanvas widget which makes it easy to write intricate, flicker-free custom displays.
- The Gnome stock pixmaps (icons for open, close, save, and other operations).
- Convenient routines for creating and using dialogs.
- The GnomePixmap widget which is more versatile than GtkPixmap.

1.2.2.3. libgnorba

libgnorba provides CORBA-related facilities, including a security mechanism and object activation. (Object activation is the process of obtaining a reference to an object that implements a given interface; it can involve executing a server program, loading a shared library module, or asking an existing program for a new object instance.)

1.2.2.4. libzvt

This small library contains a terminal widget (ZvtTerm) you can use in your Gnome programs.

1.2.2.5. libart_lgpl

This library contains graphics rendering routines by Raph Levien. The routines included here are released under the GNU Library General Public License and used in the GnomeCanvas widget; Raph Levien also sells an enhanced proprietary version. libart_lgpl provides antialiasing, microtile refresh regions, and other magic. In essence it is a vector graphics rasterization library, functionally analogous to the PostScript language.

1.2.3. Other Libraries

These libraries are commonly used in Gnome applications, but are not a part of gnome-libs proper.

1.2.3.1. gnome-print

gnome-print is still somewhat experimental, but very promising. It uses libart_lgpl and works nicely with GnomeCanvas. It provides virtual output devices (called "print contexts"), so a single codebase can output to a print preview widget, to PostScript, and eventually to other printer formats. gnome-print also includes printing-related GUI elements, like a print setup dialog, and a virtual font interface (to deal with the problem that X fonts are not printable).

1.2.3.2. gnome-xml

gnome-xml is a non-validating XML engine written by Daniel Veillard of the World Wide Web Consortium. It can parse XML into a tree structure, and output a tree structure as XML. It's useful for any application that needs to load and save structured data; many Gnome applications use it as a file format. This library does not depend on any of the others, not even glib so it is tied to Gnome in name only. However, you can expect most Gnome users to have it installed, so it should not inconvenience your users if your application uses this library.

1.2.3.3. Guile

Guile is an implementation of the Scheme programming language in a library, so that any application can have an embedded Scheme interpreter. It is the official extension language of the GNU Project, and is used by several Gnome applications. Adding an extension language to your application might sound complex, but Guile makes it almost trivial. (Several Gnome applications support Perl and Python as well; it is usually easy to support several languages once you implement the first. But Guile has a special place in the Gnome developer's hearts.)

1.2.3.4. Bonobo

At press time, the Gnome hackers were putting the finishing touches on Bonobo. Bonobo is a compound document architecture in the tradition of Microsoft's OLE; it allows you to embed charts in spreadsheets, for example. It will be used pervasively throughout Gnome; any application will be able to display MIME-typed data such as plain text, HTML, or images by asking the Gnome libraries for an appropriate Bonobo component. Look for Bonobo technology in the next major Gnome release.

1.2.4. A Word About Header Files

Throughout the book, the exact header file which declares each function is given alongside the function prototype. This is to facilitate your exploration of the source code. However, you probably don't want to manually include the hundreds of headers found in GTK+ and Gnome. You can include all GTK+ headers en masse by including the gtk/gtk.h header file. gtk/gtk.h also includes gdk/gdk.h for you. You can include all Gnome headers by including gnome.h; gnome.h includes gtk/gtk.h for you. Most Gnome application files simply include gnome.h.

1.3. Structure of the Book

This book is divided into several parts:

Right now you're reading Part 1. This chapter gives you an overview of the Gnome application development framework. The following two chapters quickly introduce glib and GTK+ – if you're already familiar with GTK+ programming, you may want to move directly to Part 2.

Part 2 guides you through the development of a skeletal Gnome application. It starts with the creation of the source tree, and then explains how to write menus, dialogs, and other essential application components. At the end of Part 2, there's a checklist of features every application should have.

Part 3 describes some advanced topics, including the internals of the GTK+ object and type system, writing a custom GtkWidget subclass, and using the GnomeCanvas widget. It also covers GDK.