1 GNUstep Frequently Asked Questions with Answers

Last updated 17 May 2015. Please send corrections to gnustep-maintainer@gnu.org. Also look at the user FAQ for more user oriented questions.

1.1 Compatibility

1.1.1 Is it easy to port OPENSTEP programs to GNUstep?

It is probably easy for simple programs. There are some portability tools to make this easier, or rewrite the Makefiles yourself. You will also have to translate the NIB files (if there are any) to GNUstep model files using the nib2gmodel program (from ftp://ftp.gnustep.org/pub/gnustep/dev-apps).

1.1.2 How about porting between Cocoa and GNUstep?

It's easier from GNUstep to Cocoa than Cocoa to GNUstep. Cocoa is constantly changing, much faster than GNUstep could hope to keep up. They have added extensions and new classes that aren't available in GNUstep yet. Plus there are some other issues. If you start with Cocoa:

- Use #ifndef GNUSTEP for Apple only code.
- Do not use CoreFoundation
- Do not use Objective-C++ (except with gcc 4.1 or later)
- Do not use Quicktime or other proprietary extension
- GNUstep should be able to read Cocoa nib files automatically, so there is no need to port these, although you might want to have GNUstep specific versions of them anyway.

See also http://mediawiki.gnustep.org/index.php/Writing_portable_code for more information.

1.1.3 Tools for porting

While the programming interface should be almost transparent between systems (expect for the unimplemented parts, of course), there are a variety of other files and tools that are necessary for porting programs.

'`nib2gmodel`'
: This program coverts nib files from any system, such as OPENSTEP to a gmodel format file. Gmodel can be read directly by GNUstep or you can convert this to a more GNUstep-native gorm format (using the Gorm interface modeller). Note this is not necessary for Cocoa nibs - GNUstep can read these directly.

'`Renaissance`'
: GNUstep Renaissance allows you to describe your user interfaces (that is, the windows in your application, and the buttons, boxes, textfields, etc in the windows) in simple and intuitive XML files, using an open, standard format describing the logic of the interface. It has a number of advantages over the proprietary nib format: portability, open standard, easy localization, themeability, and intelligent autolayout.

'`Gorm`' The equivalent of the Interface Builder in GNUstep. It might be easier to just recreate the interface using Gorm rather than dealing with translations.

'`OpenStep2GNUConverter and nfmake`'
Two programs that allow you to convert PB files to GNUstep makefiles or compile a program on GNUstep directly from PB files. They probably work only for OPENSTEP systems and are a little out-of-date.

'`StepTalk`'
A portable scripting environment that lets your do scripting in almost any language you like.

1.1.4 Can I transfer archived data from GNUstep to Cocoa?

Apple's archiving format is proprietary and not documented, so this poses a problem for anyone wanting to implement compatibility with it. However, even if we reverse engineered the format, there are enough differences between the class and ivar layouts to make this sort of compatibility difficult. Not to mention the fact that we would constantly have to keep up with the changes Apple made.

The new keyed archiving using XML file formats is much more portable, and GNUstep is trying to maintain compatibility with Apple with this type of archiving.

1.1.5 Does distributed objects work between GNUstep and Cocoa?

See the answer to the previous question (on archive compatibility) for why this won't work either.

1.1.6 Is there an Interface Builder for GNUstep?

There is an Interface Builder for GNUstep called Gorm. A lot of work has been put into it and it works very well. You can download it from the ftp site or via http. The Project Manager ProjectCenter is also available.

1.1.7 Can I use my original NIB files?

No - NeXT/Apple never documented their nib format, so GNUstep supports both the 'gmodel' format (which stores information as text (property-lists) and can therefore be edited 'by hand') and binary archive format (which can be edited by Gorm). There IS a conversion tool called nib2gmodel that can be compiled under OPENSTEP to convert nib files to GNUstep gmodel files.

The current version of gui supports reading nib files created as of 10.2. If you have nib files which are older than this, you can convert them by loading them into Interface Builder, going to the "file" second and saving the nib using the "10.2 or later format."

1.1.8 Can one use the hybrid "Objective-C++"

Yes gcc 4.1 has support for this.

1.1.9 Is there a plan to support the Java/YellowBox Bindings?

Yes. The GNUstep Java library/bridge called JIGS is available now. JIGS is a free (LGPL) Java Interface for GNUstep; it can automatically wrap Objective-C libraries based on

GNUstep, making them accessible directly to the Java programmer as if they were Java libraries. As a side effect, it is also possible to use the whole engine in the reverse way: JIGS provides a high level API to allow Objective-C programmers to start java virtual machines inside GNUstep Objective-C code and access java objects in the java virtual machine transparently, as if they were objective-C objects.

1.1.10 What if I compile GNUstep under OPENSTEP/MacOS X?

GNUstep uses different backends to provide the same functionality as Display Postscript. While someone could write a backend library to provide the interface, nobody has bothered to date.

You can, however, use a GNUstep program with an X11 server running on MacOSX.

1.1.11 Is the Objective C API for GTK related?

No. GNUstep applications provide their GUI via the OpenStep API, which provides fully object-oriented access to GUI manipulation.

The object-oriented nature of the libraries and language make it much easier for new users to create their own subclasses rather than simply using the supplied widgets as in other frameworks.

1.1.12 How about implementing parts of the Application Kit with GTK?

Yes and No - The GNUstep architecture provides a single, platform-independent, API for handling all aspects of GUI interaction (implemented in the gstep-gui library), with a backend architecture that permits you to have different display models (display postscript, X-windows, win32, berlin ...) while letting you use the same code for printing as for displaying. Use of GTK in the frontend gui library would remove some of those advantages without adding any.

That being said, a backend library could be implemented using gtk if anyone wanted to do so. Since the frontend library handles most of the work involved in implementing the OpenStep API, the backend is a relatively thin layer and the advantages of GTK over direct xlib or win32 calls is likely to be minimal. If/when GTK is ported to more systems, a backend written using it could be a valuable asset - volunteers are, as always, welcome.

1.2 Compiling and Developing

1.2.1 How can I get started programming?

Good question. Read the tutorials at the GNUstep web site. Also look at Apple's documentation (pointers in the Resources section on the GNUstep web site.)

1.2.2 How can I help with GNUstep?

1. Write/debug library code
2. Write documentation
3. Update the task list and library headers
4. Write applications

Let people know what you are doing. Break your project up into the smallest units you can. Feed back frequent updates to the maintainers. Ask questions in the discussion mailing list.

Do remember that any changes beyond a few lines of code (or documentation) require a disclaimer or copyright assignment to the Free Software Foundation before they can be incorporated into the project. Get in touch with the GNUstep maintainer about this.

Don't start with large-scale reorganization of anything - instead, get a general idea in mind of what you want to do, and proceed as much as possible with incremental changes that don't break anything - that way you can make those incremental changes available to the rest of the community at frequent intervals.

Don't be afraid to give up - there is no shame in finding out that you have take on too large/complex a project. It's much better to 'resign' and take on a smaller job than to just stop without telling anyone.

Please document the code you add or change (using autogsdoc comments that begin with a slash and two asterices). But PLEASE, do not copy from the Apple documentation or any other copyrighted documentation.

1.2.3 Helping develop GNUstep

There is plenty of unimplemented stuff in the gui library and backend libraries that volunteers can work on - just browse through the code and see if it conforms to the documentation.

Specific tasks are noted in the developers section on the GNUstep website.

Once you have coded something, you could always write a testcase and documentation for it :-)

1.2.4 Helping document GNUstep

All class documentation is written directly in the source code itself and translated using the autogsdoc program. See the source code and documentation for autogsdoc for information on documenting the classes.

Newcomers could write documentation for individual classes by comparing the OpenStep specification, the MacOS-X documentation, and the GNUstep source. Documentation should clearly note where individual methods are specific to OpenStep, MacOS-X or are GNustep extensions.

More experienced people could write documentation on general programming topics, and tutorials for new users.

Anyone willing to write documentation, either tutorials for using GNUstep, or reference documentation for individual classes, should either write it in gsdoc or as plain ascii text for someone else to format into gsdoc.

GNUstep documentation should have copyright assigned to the Free Software Foundation.

1.2.5 How do I assign my contribution?

Everyone who contributes more than 20 lines of code or so needs to sign a copyright assignment so that the FSF can have legal control of the copyright. This makes it easier to defend against any copyright infringement suits. Contact the GNUstep maintainer for instructions

on how to do this or download and fill out the form http://www.gnustep.org/resources/request-assign.future (instructions are included).

1.2.6 How do I update the task list?

The task list (http://savannah.gnu.org/pm/?group_id=99) is supposed to tell people what jobs are waiting to be done. Feel free to add to it or update the tasks that are there (you need to create a login for yourself first).

One job of major importance that pretty much anyone can do is to look for jobs to add to the task list. In the case of methods from the OpenStep specification or the MacOS-X documentation not being present in the GNUstep libraries, it is also helpful to add the method prototypes to the library header files.

Send any changes or additions to bug-gnustep@gnu.org.

A beginner can look through the MacOS-X documentation, the OpenStep specification and the GNUstep source and contribute task items.

If a class or method is in MacOS-X and OpenStep but is not in GNUstep - it's a high priority TODO and should at least be added to the GNUstep headers and a dummy version added to the source with a FIXME comment.

If a class or method is in MacOS-X but not OpenStep or GNUstep - it's a low priority TODO. It should be added to the GNUstep headers bracketed in `#ifndef STRICT_OPENSTEP`

If a class or method is in OpenStep but not in MacOS-X or GNUstep - it's a low priority TODO. It should be added to the GNUstep headers bracketed in `#ifndef STRICT_MACOS_X`

There are a couple of people working on this already, so it's a good idea to get in touch with Greg, Adam or Richard to coordinate efforts.

1.2.7 How do I start writing tests?

You can write testcases - where the libraries fail tests, you could either fix the problem, or add it to the task list.

To write testcases, you need to use svn to install the latest GNUstep sourcecode you can find. Then checkout the 'gnustep/tools/testsuite' module from svn.

1.2.8 How do I start writing applications?

You can either look at the links on the GNUstep website for applications that have been started, and email their owners to volunteer to help, or you can start your own project.

1.2.9 How can I help with the GNUstep website?

Talk to Adam Fedor fedor@gnu.org, the maintainer.

The GNUstep website is kept as a CVS module, but the largest portions of it (the FAQ and the Documentation) are actually generated from files in the individual GNUstep packages. Many highly changeable portions are kept on the Wiki, so anyone can change these (first you need to get write access, though).

If you want to update the FAQ or documentation - grab the latest snapshot of the GNUstep core you can find, update it from the svn repository, and work with the contents of the appropriate documentation directory.

If you want to work on other parts of the website, you can grab a copy of the website via anonymous CVS. See http://savannah.gnu.org/cvs/?group_id=99 for instructions on how to do that.

The main task with the website is to figure out which bits are out-of-date (or wrong) and update/mark-as-outdated as required.

1.2.10 Why doesn't GDB support Objective-C?

As of GDB 6.0, gdb supports debugging of Objective-C code.

1.3 GNU Objective C Compiler and Runtime

1.3.1 What is the Objective C Runtime?

The Objective C Runtime Library provides C functions and data structures required to execute an Objective C program.

The GNU Objective C Runtime Library offers everything NeXT's runtime does, including Categories, Protocols, '`+poseAs:`', thread-safety, class initialization on demand, delayed loading of classes, and initialization of static instances (such as @""-style string objects).

It also has several differences over NeXT's implementation:

- GNU's runtime provides "selector-types" along with each selector; NeXT's does not. A selector-type is a string that describes the C variable types for the method's return and argument values. Among other uses, selector-types is extremely helpful for fast distributed objects implementations, (see GNUstep Base Library Section, below).
- Many of the GNU functions have different names than their corresponding NeXT functions; the GNU names conform to the GNU coding standards. The GNUstep base library contains a compatibility header that works with both runtimes. You should use functions there or use OpenStep Foundation methods/functions instead of the basic runtime functions so that you code can run with either system.

  Apple has recently added new functionality to their runtime, including built-in exception handling, etc. Hopefully these will be ported to the GNU runtime in the future.

1.4 GNUstep Base Library

1.4.1 What is the GNUstep Base Library?

The GNUstep Base Library is a library of general-purpose, non-graphical Objective C objects. For example, it includes classes for strings, object collections, byte streams, typed coders, invocations, notifications, notification dispatchers, moments in time, network ports, remote object messaging support (distributed objects), and event loops.

It provides functionality that aims to implement the non-graphical portion of the OpenStep standard (the Foundation library).

1.4.2 What is its current state of development?

GNUstep base is currently stable and, to the best of our knowledge, implements all of the OpenStep functionality (except for a few classes that we feel are not useful). It also implements most all of the new Cocoa classes. However we do some things, like scripting, differently, so we don't implement all the Cocoa classes.

1.4.3 What are the features of GNU Distributed Objects?

GNU Distributed Objects has many of the features of other distributed objects implementations, but, since it is free software, it can be ported to platforms for which other distributed objects implementations are not available.

[NOTE: The GNU distributed object facilities have the same ease-of-use as Apple's; be warned, however, that they are not compatible with each other. They have different class hierarchies, different instance variables, different method names, different implementation strategies and different network message formats. You cannot communicate with a Apple NSConnection using a GNU NSConnection.

Here are some differences between GNU distributed objects and Apple's distributed objects: Apple NSDistantObject asks it's remote target for the method encoding types and caches the results; GNU NSDistantObject gets the types directly from the local GNU "typed selector" mechanism if the information is known locally and only queries the remote target or caching encoding types when using a method that is not known to the local process. The NSProxy for the remote root object always has name and, once set, you cannot change the root object of a NSConnection; the GNU Proxy for the remote root object has a target address value just like all other Proxy's, and you can change the root object as many times as you like.].

1.5 GNUstep GUI Library

1.5.1 What is the GUI Library?

The GNUstep GUI Library is a library of objects useful for writing graphical applications. For example, it includes classes for drawing and manipulating graphics objects on the screen: windows, menus, buttons, sliders, text fields, and events. There are also many peripheral classes that offer operating-system-independent interfaces to images, cursors, colors, fonts, pasteboards, printing. There are also workspace support classes such as data links, open/save panels, context-dependent help, spell checking.

It provides functionality that aims to implement the '`AppKit`' portion of the OpenStep standard. However the implementation has been written to take advantage of GNUstep enhancements wherever possible.

1.5.2 Explain the organization of the front- and back-ends

The GNUstep GUI Library is divided into a front- and back-end. The front-end contains the majority of implementation, but leaves out the low-level drawing and event code. A back-end can override whatever methods necessary in order to implement low-level drawing event receiving. Different back-ends will make GNUstep available on various platforms. The default GNU back-end will run on top of X Windows. Other back-ends could allow GNUstep to run on OpenGL and WIN32 graphics/event platforms. Much work will be saved by this clean separation between front- and back-end, because it allows different platforms to share the large amount of front-end code.

1.5.3 What is the current state of development of the front-end?

Many of the classes are well implemented, if not thoroughly tested. See the GNUstep web sites and read status information contained in the distribution for the most up-to-date information.

1.5.4 What is the current state of development of the back-ends?

There are several backends currently available:

'`xlib`' This backend runs on X11 and uses standard xlib calls for implementing drawing. It works well, but is limited in many areas due to the limitations of xlib drawing.

'`art`' This is a very good backend that draws using the libart package and freetype with near PostScript quality and functionality. It is currently the standard backend (as long as the required libraries are installed).

'`w32`' This backend works on Windows and uses basic Windows drawing

'`cairo`' An up-and-coming backend. It still relies on unpublished functions in the cairo library so using it is not for the beginner.

1.6 GNUstep DisplayGhostScript Server

1.6.1 What is the Display Ghostscript Server?

It is a free implementation of a Display PostScript server based on the GNU Ghostscript program developed by Aladdin Enterprises and now owned by artofcode LLC.

At one point, GNUstep was using this for display purposes. However the development of DGS has stopped as it is too difficult to maintain and no one wanted to work on it. Now we are using other means of drawing.

1.6.2 What is its current state of development?

GNU contracted with Aladdin Enterprises to add some key features to GNU Ghostscript so it could be used as a DPS server. This work has mostly been done, although Aladdin did not completely finish the work that they were contracted for. (Because the work took longer than specified and was not completed, Aladdin agreed to waive approximately $10,000 in promised fees for the work that was actually done and delivered.) DGS works fairly well with a single context. Alpha channel and compositing doesn't work.

Currently, further development on DGS has been abandoned. The library based approach using libart, cairo, etc works much better.

1.6.3 What is the relationship between the Display Ghostscript Server and X Windows?

Display Ghostscript runs on top of X Windows.