GNOME Frequently Asked Questions

Todd Graham Lewis

David "Gleef" Zoll

Julian "X-ViRGE" Missig

GNOME Frequently Asked Questions

by Todd Graham Lewis, David "Gleef" Zoll, and Julian "X-ViRGE" Missig

v1.0.7 Edition 

This is the FAQ for GNOME, the GNU Network Object Model Environment.

Table of Contents

Chapter 1. GNOME Overview

This section covers questions that pertain to GNOME as a whole.

What does GNOME stand for?

GNOME stands for "GNU Network Object Model Environment". GNU stands for "GNU's Not Unix".

What is GNOME?

GNOME is the GUI desktop of the GNU Project (http://www.gnu.org).

To quote from the original announcement from comp.os.linux.announce, GNOME is intended to be "a free and complete set of user friendly applications and desktop tools, similar to CDE and KDE but based entirely on free software."

I, your lowly FAQ maintainer, like to explain GNOME as everything that's expected in a modern programming environment. In this respect, it is approximately equivalent to CDE, Win32, NextStep, or KDE. The big difference is that, unlike any of the above-mentioned examples, every single component of GNOME is Free Software (http://www.gnu.org/philosophy/free-sw.html). Not only that, but GNOME is extremely customizable compared to most desktop environments.

For the project geographers out there, here are some of the higher mountains to be found in GNOME-land:

- GNOME uses the Object Management Group's Common Object Request Broker Architecture to allow software components to inter-operate seamlessly, regardless of the language in which they are implemented, or even what machine they are running on! (See CORBA)
- The GNOME community is working hard on developing an object model called Bonobo. Based on CORBA and similar to Microsoft's Object Linking and Embedding, v2 (OLE2), Bonobo will allow programmers to export and/or import componentized resources. This, for example, would allow users to use whatever editor they like in their development environment, provided that their editor supported via CORBA a standardized editor interface (See Bonobo)
- GNOME is not tied to any one window manager. You can chose your favorite window manager and use it with GNOME. (See Window Manager)
- GNOME uses the GIMP Tool Kit (GTK+) as the graphics toolkit for all graphical applications. GTK+ has tons of neat features, but my three favs are: (1) support for multiple languages, including C, C++,

Objective-C, Scheme, Perl, and others; (2) themability, where a user can change the look and feel of all GTK+ apps on a machine while apps are running, which is just unbelievably cool; (3) and finally, GTK+ is licensed under the LGPL, meaning that it is completely Free Software, just like the rest of GNOME. (See GTK+)

- Along with GTK+, GNOME uses Imlib, an image library for the X Window System which supports multiple image formats, from XPM to PNG, and multiple bit-depths, from 24-bit True Color down to 1-bit Black and White, all transparently to the programmer. (See Imlib)
- All GNOME applications are session aware. This means, e.g., that if you shut down the GNOME word processor and then start it back up again, it will open up the document which you had open before, and put your cursor back at the same place. This is accomplished via the X Session Management system, as implemented in the GNOME Session Manager. (See Session Management)
- GNOME uses the DocBook SGML standard for all documentation, which allows programmers to write documentation in a straightforward way. Documentation can then be viewed with the GNOME help browser, or rendered into HTML and viewed with a web browser, or converted into LaTeX or postscript and printed out. (See SGML)
- GNOME supports the Uniforum standard internationalization and localization methods, allowing support for new languages to be added without even requiring a recompile of the application. (See Internationalization & Localization)
- GNOME applications will support several Drag and Drop protocols for maximum interoperability with other applications. (See Drag and Drop)
- GNOME will support 3D programming with OpenGL, allowing people to use 3-dimensional graphics right in their GNOME apps, whether they be scientific applications or games. GNOME uses Mesa, which is a free software implementation of the OpenGL standard. (See OpenGL)

If all of this seems ambitious, that's because it is! Read on to find out more!

Where can I find more information, help, or tips on about GNOME?

The GNOME home page is, of course a good place to start: http://www.gnome.org.

You also should consider joining the gnome-list mailing list, it is a very good place to ask for help.

How do I use these GNOME mailing lists I keep hearing

about?

The best place for up-to-date help and information on GNOME is the mailing lists. You can find them listed at http://www.gnome.org/mailing-lists/index.shtml. The general-purpose list for help and discussion is gnome-list@gnome.org (mailto:gnome-list-request@gnome.org). To join this (or any) of the lists, send email to the *request address*, this is generally the list address with "-request" added to the name. For example the request list for gnome-list is gnome-list-request@gnome.org (mailto:gnome-list-request@gnome.org). The subject of the email *must* be **`subscribe`**.

For questions about developing GNOME applications and developing GNOME, suscribe to the gnome-devel-list@gnome.org (mailto:gnome-devel-list-request@gnome.org) mailing list.

If you have a good idea or a great concept on how to improve the user interface experience in GNOME, please subscribe to the gnome-gui-list@gnome.org (mailto:gnome-gui-list-request@gnome.org). The user interface team is lead by Jim Cape who maintains a web page with the suggested improvements at http://www.jcinteractive.com/gnome-ui/

Within a few minutes (possibly a little longer if you aren't in North America), you should get an automatic reply back asking for confirmation. Follow the directions in this email, and you will be subscribed. Finally, the system will send you two emails with help on how to use the system.

How do I get to the mailing list archives?

There are two ways. The easiest is to go to the mailing list page at http://www.gnome.org/mailing-lists/index.shtml. Click on any of the list addresses on the left, you will be able to browse the archives of that list.

If you need to search the archives, you can use the request address for that list. One of the emails it sent you when you first subscribed gives instructions on how to do this. If you have lost those instructions, email gnome-list-request@gnome.org (mailto:gnome-list-request@gnome.org) with a subject of **`archive help`**. It will reply with instructions on how to use the archive system.

How do I pronounce GNOME?

GNOME stands for "GNU Network Object Model Environment". GNU stands for "GNU's Not Unix", and has always been officially pronounced "guh-NEW" to minimize confusion. Since GNU is GNOME's first name, GNOME is officially pronounced "guh-NOME".

However, many people pronounce GNOME as just "NOME" (like those short people from legend), nobody will hurt you if you find this pronunciation easier.

Why GNOME?

First, the GNU operating system needs a desktop environment that is 100% free software. GNOME satisfies this need.

The second goal, by way of the first goal, is that people who use free operating systems will have a nice user environment in which to work (This works also on other non-free UNIX-based systems).

Finally, GNOME has a lot of really neat features that geeks are guaranteed to enjoy. Says one reviewer of this FAQ:

> Todd, GNOME has technical merits even beyond "open source" and "nice environment" and "world domination," as you well know...For instance, the COM clone and the use of CORBA for inter-operation gives me a woody.

This is certainly true (the fact that GNOME has great technical merit; I have no knowledge on the woody front.) Try it out for yourself and see!

Why not CDE? or KDE? or GNUStep?

Because GNOME is better! Seriously though, each of these desktop systems have issues which encouraged us to start fresh with the GNOME project.

CDE is not Free Software, it isn't even close. Many people also find it lacking in features, performance and functionality.

At the time of GNOME's inception, KDE had serious licensing problems, which they are still trying to resolve. First, KDE is based on a library called Qt, which was originally non-free. However TrollTech's recent release of Qt version 2.0 uses the QPL, so it qualifies as Free Software. KDE Development releases are based on the QPL'ed versions of Qt.

The other issue is that originally, KDE programs were released under the GPL. The GPL was designed to encourage more GNU software, so it doesn't work well when linked to software under other licenses. This incompatibility doesn't prevent anyone from using KDE, but it does make it troublesome to distribute. For more details, you can read Debian's position on KDE (http://www.debian.org/News/1998/19981008). KDE is in the process of changing their licenses over to the Artistic license, which avoids the problem. However, changing licenses on a large project with many authors like KDE is a long and difficult process, so it will take them a while.

The GNOME people like the KDE people, and we consider this an unfortunate situation that is in the process of being fixed. Hopefully, this will cease to be an issue soon, and GNOME and KDE can compete friendlily on technical merit and design. This matter has been hashed out time and time again on the gnome-list mailing list. Asking this question on gnome-list is discouraged behavior. If you want to go

somewhere and start a flame war on this topic, then please do it somewhere far far away where we don't have to listen to you.

There are significant design differences between KDE and GNOME. Top of the list is a difference in widget set. We find GTK+ to be nicer, more customizable, more friendly to development in various programming languages, and more flexible than Qt; others may disagree. Also near the top of the list is the fact that GNOME is not tied to any single window manager. You don't have to use Enlightenment, you can use any GNOME-compliant window manager. In all, the projects are different enough so that both should be able to coexist and even collaborate.

GNUStep is another desktop environment that has a lot of good things going for it. One issue with GNUStep are that they are trying to reimplement the OpenStep desktop and API, with GNOME we'd rather develop something new and good rather than just redo something old, even something old and good. Also, just like Qt strongly pushes C++ on KDE developers, GNUStep strongly pushes Objective C on its developers. GNOME wants to remain language agnostic, and support development in whatever language you wish to use.

What platforms?

GNOME was started by several people well-known in the Linux and GNU communities, but it is intended to run on any modern and functional Unix-like system. GNOME has been reported to work under the following:

GNU/Linux
BSD (FreeBSD, NetBSD and OpenBSD)
Solaris
IRIX
HP-UX
AIX
probably some others; I have not really kept track.

If you try GNOME on other platforms and it works, then please be sure to let me know so that I can include it here. If you try it on another platform and have trouble, then please report the difficulties which you encounter, so that the GNOME developers can work on the problem. (See Reporting Bugs.)

Can I still run my programs under GNOME?

GNOME doesn't replace your system, it runs on top of it. All programs that run without GNOME should run fine alongside GNOME.

What is a window manager, and what does it have to do with GNOME?

A window manager is a program within the X Window System environment which controls the placement and appearance of windows on the screen. Look at an xterm or something. The box of terminal text is the actual xterm, the border, title bar, buttons and so forth are all handled by the window manager. Some window managers have extra features, but all handle window placement and decoration.

There are many window managers out there, and GNOME should work well with all of them. There are a few GNOME features (such as session management, and the GNOME pager) that won't work with all of them. For best results, go with a GNOME-compliant window manager. The most compliant are currently Enlightenment, Sawmill, Window Maker, and IceWM, with FVWM, SCWM, AfterStep and QVWM putting finishing touches on GNOME-compliance as we speak.

You've said that GNOME is Free Software. What does Free Software mean?

Most software licenses are designed to limit the users freedom to use, examine, modify and distribute the software. In the GNOME project, and the Free Software community in general, we believe that such restrictions are unethical, and we endeavor to produce quality software that is free of such restrictions. This is what we call Free Software. Traditionally, it has referred to software which allows three types of freedom:

- The freedom to study how the program works and adapt it to your needs.
- The freedom to redistribute copies so you can share with your neighbor.
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.

More recently, a fourth freedom has started being restricted by licenses, the freedom to use the software. Since software licenses are based on copyright law, and copyright law only addresses distribution, not use, we assumed that freedom of use goes without saying. Suffice it to say that while freedom of use is not part of the traditional definition of Free Software, the concept does include that freedom.

* *Note to translators, please translate the following note, including all instances of the word "free" carefully, except leave the* `free` *in English.*

> **Note:** If the following paragraph is hard to understand in translation, forgive me. Unlike most languages, in English the word used in both "free beer" and "free speech" is `free`. This causes some confusion when discussing Free Software.

Free Software is a matter of liberty, not price. To understand the concept, think "Free Speech" not "Free Beer". The Free Software community is not at all against the concept of selling copies of software, or programmers making a living. We are not against commercial software, we are against proprietary software. For more information, check out the articles at http://www.gnu.org/philosophy (http://www.gnu.org/philosophy).

In the GNOME project, we protect the user's freedom by using the GNU General Public License (GPL) and the GNU Lesser General Public License (LGPL, formerly known as the GNU Library GPL). These licenses are carefully designed with legal advice to produce a "copyleft". That is a copyrighted work that guarantees that it will always be freely redistributable. This is by making sure that modifications and derivative works are also covered under the GPL.

The main GNOME applications (and a few of the libraries) are covered under the GPL. Most of the core GNOME libraries use the LGPL, to allow other free (and proprietary) software licenses to be used when making GNOME software, while still protecting the libraries from being restricted.

Is GNOME Open Source(tm) Software?

The short answer is yes. The GPL and LGPL used for GNOME are explicitly listed as examples for the "Open Source" definition. However, we try to avoid using the term Open Source to describe GNOME, since not only does the term deliberately hide the freedom inherent in Free Software, but the term is being used for software which is not Free.

The term was developed to try to bring Free Software to businesses who didn't want to hear about Freedom. Personally, I have found that most businesses like to hear about freedom. The freedom to customize their software to their needs, the freedom to change software developers without changing their software, the freedom to give copies of their software to their clients and subcontractors, these are all freedoms that Free Software offers, and proprietary software usually doesn't, that make *good business sense*. For more on the distinction between Open Source and Free Software, see the article by Richard M. Stallman at http://www.gnu.org/philosophy/free-software-for-freedom.html (http://www.gnu.org/philosophy/free-software-for-freedom.html).

Chapter 2. Getting GNOME

There are many ways to get GNOME on your system. This section covers the most common questions involved in the process.

What are the System Requirements for GNOME?

Currently, you need a machine with Unix or a Unix-like operating system installed, with the X Window System (X11R5 or later). You need at least 16MB of RAM, although I expect you'll be happier with 32MB or more. If you are just installing binaries, you need about 30MB of disk space, count on needing at least 200MB of disk space if you are compiling from source.

What's the best format to get GNOME for my machine?

It depends on what kind of computer you have, and what you want to do with GNOME. If in doubt, the officially supported way to install GNOME is from the latest set of tarballs (the files ending in ".tar.gz") from ftp://ftp.gnome.org or its mirrors. Look in the /pub/GNOME/sources directory for the various packages. The full list of FTP mirrors is at http://www.gnome.org/ftpmirrors.shtml, in an emergency, try ftp://gnomeftp.wgn.net/pub/gnome.

If you want the up to the minute, bleeding edge development version of GNOME, CVS is the way to go. There is no guarantee that the CVS version will even compile, much less work, but bug reports are quite appreciated. If this sounds like its for you, you can find more information on getting the CVS version at http://www.gnome.org/devel/whatiscvs.shtml and http://www.cinternet.net/~soren/gnome/cvs.shtml.

If you don't want to deal with compiling GNOME, and you are on a common system, there might be precompiled binary packages available for you. Currently there are packages for RedHat, SuSE 6.0, Debian and LinuxPPC systems. Until recently there were Solaris packages as well, as soon as we have good ones we will offer these as well. Find out more about the various distributions at http://www.gnome.org/start/getting.shtml.

What do I need on my computer before I install GNOME?

There are a few things that GNOME counts on your system having, if you don't have them, or have an older version, install or upgrade the following packages before you start the process of installing GNOME.

libjpeg (6b or higher)
libpng (1.0.1, 1.0.3 or higher *not 1.0.2*
libungif (3.0 or higher)
libtiff (3.4 or higher)
libgr (2.0 or higher)
ImageMagick (4.0.5 or higher)
zlib (1.1.2 or higher)
guile (1.2 or higher)
Berkeley DB (1.85 or higher)

Some of the bells and whistles require a GNOME-compliant window manager to work. Currently these are Enlightenment, IceWM, Sawmill, and Window Maker. In addition both FVWM and SCWM are putting the finishing touches on their GNOME support. GNOME does not include a window manager, and will work well with any window manager, these just give you a few extras.

The GNOME libraries require some packages, which are:

audiofile - Audio file format library
esound - Sound server
glib - Utility routines
libxml - XML library
ORBit - CORBA implementation
GTK+ - Widget set
imlib - Image loading and manipulation library

The main GNOME modules require:

gnome-libs - The main GNOME libraries
libgtop - Portable system status access library
libghttp - HTTP access library

And now we have reached the main GNOME modules. You'll probably want these:

gnome-core - Panel, help browser, session manager
mc (gmc, mc, mcserv) - File manager, desktop icons
control-center (gnomecc) - Graphical configuration for user settings

There are also some packages that are needed by a few, non-critical programs within GNOME. If you are running short on space or time, you can safely skip these. If you want all the bells and whistles to work, you will want these:

libglade - needed for some apps
gnome-print - needed for some apps
ee - Image viewer
gtop - CPU & memory usage monitoring
gnome-media - CD player, volume control, etc

gnome-pim - Calendar, address book
gnome-utils - Hex editor, system info, file finder
gtk-engines - GTK+ themes
xchat - IRC client
gnumeric - Spreadsheet
gnome-audio - Sound files
gnome-games - assorted games
users-guide - GNOME User's Guide

Lastly, there are several packages of development tools. Note that these are different from -devel RPMs, which you DO need if you plan on compiling GNOME applications at all. If you plan on doing development with these tools/libraries, download them:

glade - A GUI builder
gnome-python - Python language bindings for GNOME
Gtk– - C++ language bindings for GNOME

What order should I deal with the GNOME packages?

There is no one answer to this question, but here's an order that should work well:

audiofile
esound
glib
libxml
ORBit
GTK+
imlib
gnome-libs
libgtop
libghttp
gnome-core
mc (gmc, mc, mcserv)
control-center (gnomecc)
libglade
gnome-print
And then anything else

How do I get GNOME tarballs?

Tarball (i.e. compressed archives of the source code using tar and gzip) releases of GNOME are made at ftp://ftp.gnome.org under the /pub/GNOME/ (ftp://ftp.gnome.org/pub/GNOME) directory. These files are then copied to the GNOME FTP mirror sites, listed at http://www.gnome.org/ftpmirrors.shtml. If both the ftp and web site are down, a relatively safe mirror is at ftp://gnomeftp.wgn.net/pub/gnome.

These tarballs are the only official releases of GNOME, and the only version that the developers can be guaranteed to have direct control over. Currently, the most recent versions of each package can be found in sources/latest (ftp://ftp.gnome.org/pub/GNOME/sources/latest), the current grouping of tarballs for the GNOME-1.0 release are in gnome-1.0/sources (ftp://ftp.gnome.org/pub/GNOME/gnome-1.0/sources).

If you want up-to-date development tarballs of GNOME, Jim Pick Software (http://jimpick.com) posts daily snapshots of the development tree in tarball form. You can get them at ftp://ftp.jimpick.com/pub/gnome/snap. Generally he keeps a few key days of each package on file, and diff files so that you can recreate the snapshots of any other day. Note that these are not releases, they are unstable snapshots, and should not be used in a production system unless you know exactly what you are doing.

How do I get GNOME from CVS?

CVS is a very useful tool, not only for developers, but for users who want to have up-to-the-second access to the development tree. However, using CVS is a little involved for this section of the FAQ. Two good places to start are the “What is CVS?” section of the website at http://www.gnome.org/devel/whatiscvs.shtml, and the Becoming a GNOMEr section below.

My only access to the internet is email. How do I get GNOME?

If you are so disconnected from the internet that you only have email access, there are remailers out there that allow you to use FTP via email. Send a message with a body of **help**, with no quotes or punctuation to bitftp@pucc.princeton.edu or webmail.ucc.ie. It may be slow, but it will allow you to download GNOME tarballs via email. Check out http://www.csdl.tamu.edu/~cchung/cpsc689/ftpmail/ftpmtoc.html or http://www.cix.co.uk/~net-services/mrcool/stats.htm for an up to date list of available FTPMail sites.

The FTP sites you would be accessing through FTPMail are detailed above in “How do I get GNOME Tarballs”, you would just use the FTPMail interface rather than directly FTPing to the site.

I’m running RedHat Linux, how do I get RPMs?

Periodically, RedHat Labs produces RPMs of the GNOME releases for RedHat Linux, and they get put on the website. The latest version is currently in gnome-1.0/redhat (ftp://ftp.gnome.org/pub/GNOME/gnome-1.0.53/redhat). They are built for alpha, sparc and i386 platforms. Make sure you get the RPMs from the noarch directory as well, whichever platform you are on. Step-by-step instructions for these RPMs can be found at http://www.gnome.org/start/getting.shtml.

These RPMs should also work fine for Mandrake Linux, and the MacMillian and CheapBytes RedHat distributions as well. These RPMs are likely to have problems with Caldera, SuSE, and other RPM-based distributions; read below for more help for these systems.

These RPMs are compiled for sound support, which is great unless you have no sound card on your system. If someone wants to maintain a RedHat Linux RPM distribution compiled *without* sound support, we will gladly put it on FTP as well.

I’m running Debian GNU/Linux, how do I get .deb packages?

* *Thank you Havoc! :-)*

The most up-to-date Debian packages are distributed on the Debian sites (and mirrors) in debian-unstable (http://www.debian.org/Packages/unstable/x11/). Step-by-step instructions are at http://www.gnome.org/start/getting.shtml.

To get the Debian packages, you just do the following:

```
$ apt-get update
$ apt-get install gnome-control-center gnome-panel gnome-session gnome-
games gnome-terminal gnome-utils
```

The Debian package maintainers are volunteers, and they try to test things thoroughly before releasing, so if you don’t see the latest release in deb format right away, please be patient. Better yet, if you want to help prepare the Debian packages for distribution, there is a mailing list at debian-gtk-gnome@lists.debian.org (mailto:debian-gtk-gnome@lists.debian.org), and instructions on how to access the staging area at http://www.debian.org/~jim/debian-gtk-gnome/README.

Since the Debian GNOME packages are already being distributed with Debian, we don’t currently mirror them here. There doesn’t appear to be any need to. If there is a real need for us to mirror these packages, just let us know and we will.

I’m running SuSE Linux, how do I get RPM packages?

There is an excellent GNOME for SuSE page at http://gnome.linuxbe.org(Belgium) and http://www.tu-harburg.de/skf/Pub/ifmpc118.ifm.uni-hamburg.de/gnome.ht(Germany). They currently have instructions and RPMs for SuSE v6.0, with v5.3 RPMs coming soon. Check it out for more information.

I’m running Caldera OpenLinux, or some other RPM-based Linux distribution, how do I get RPM packages?

Just because two distributions use RPMs doesn’t mean they can use the same RPMs, for example, RedHat and Caldera use different versions of libc, so RedHat binary RPMs probably won’t work on Caldera. One trick that might work is to take the SRPMS from the RedHat distribution, and use **rpm –rebuild** on them, and then install the resulting binary RPMs. Otherwise, you probably have to follow the tarball instructions.

I’m running FreeBSD, how can I get GNOME via Ports?

* *Thank you Stanislav Shalunov! :-)*

FreeBSD has ports and packages for various GNOME applications (including Gnumeric) as well as a port for the entire GNOME project that will install all GNOME-specific programs via dependencies. If you have the CDROM mounted under `/cdrom` (the name of the mountpoint *is* important) or you have a working internet connection and the time to download for a while, you can install GNOME on your FreeBSD box (3.1 or later, possibly early as well) by typing:

```
$ cd /usr/ports/x11/gnome && make install
```

To learn more about the FreeBSD Ports Collection you can visit http://www.freebsd.org/ports. If you don’t have an internet connection, but do have a docs distribution, you can also find more information at `/usr/share/doc/handbook23.html`. This document will also tell you how to install GNOME as pre-built packages (binaries) using **pkg_add**, if you have the CDROM. Currently (for 3.1-RELEASE), you will need to install the following packages: `gnomelibs`, `gnomeaudio`, `gnomecore`, `gnomegames`, `gnomeprint`, `gnumeric`, and `libgtop`. A listing of the latest versions available can be found at http://www.freebsd.org/ports/gnome.html.

I’m running Solaris, AIX, HP-UX, BSDI, SCO, IRIX, or some other Unix or Unix-like system, can I get binary packages?

There used to be binary packages for Solaris that someone made, but the packages seem to have disappeared (it was an old version of GNOME anyway). If a good set of binary packages for any of these systems becomes available, we can probably put them on the FTP site.

In absence of a binary version, you’re going to need to use the tarballs. If your system doesn’t have a compiler, the GNU Project (http://www.gnu.org) offers the GNU C Compiler (gcc) for Free on pretty much every platform out there. Go to http://www.gnu.org/software/gcc/gcc.html for more information.

I’m running Microsoft Windows, can I get GNOME?

* *Thank you David Wheeler!*

Currently the answer is no, GNOME hasn’t been ported to the Windows platform, there is an effort going on to make such a port possible. If you really really need a version for Windows, count on doing lots of work and coordinating with related efforts.

Tor Lillqvist has been porting several graphical libraries used by GNOME to the Win32 (Microsoft Windows 95/98/NT) platform. More info on his efforts can be found at http://www.gimp.org/~tml/gimp/win32. He’s already ported glib, gtk+, gdk_imlib, and a few other libraries involved in generating graphics to Win32.

His goal is to get Gimp working fully on the platform (it already works partially), but his work brings GNOME much closer to being portable to Windows. A number of GNOME applications need just his libraries (possibly with some more debugging), libpng, gnome-libs and ORBit. If someone were to port these, many GNOME applications could be compiled and run on Windows.

If you are interested in working on such a project, make sure to carefully read the documentation of Autoconf (http://www.gnu.org/software/autoconf/autoconf.html), since that’s how we differentiate between different system capabilities in GNOME. Also, take a close look at Cygnus Solution’s CygWin32 (http://sourceware.cygnus.com/cygwin/) libraries and utilities. They will make such a task much easier.

Chapter 3. Compiling GNOME

What order do I compile the packages in?

It is important that you compile and install the GNOME packages in the proper orders. There are some serious dependencies within the GNOME tree. Here is my suggested order, you should compile and install each package before you move onto the next. On ELF-based systems (such as GNU/Linux and FreeBSD), you should also run **ldconfig** after each package, just to be sure any new libraries are found.

audiofile
esound
glib
libxml
ORBit
GTK+
imlib
gnome-libs
libgtop
libghttp
gnome-core
mc (gmc, mc, mcserv)
control-center (gnomecc)
libglade
gnome-print
And then anything else

How do I configure the source code for my system? (or... Where are the Makefiles hiding?)

GNOME uses configure (generated by Autoconf) to set up the source code to compiler properly on your system. Configure will investigate how your system is set up, and define compiler constants to add or remove various pieces of code as needed.

Configure accepts a variety of options, type **./configure –help** for the full list. The most important one is **--prefix**, which tells the system where GNOME is to be installed. If you omit the prefix, configure assumes you want `/usr/local`. Most RPMs use a prefix of `/usr`. Many users prefer to use a prefix of **`/opt/gnome`**, which requires a little advance preparation described below.

Configure also checks certain environment variables. Set your CFLAGS to equal whatever compiler options you want to make sure get passed, for example **`-O6`** will maximize optimization, **`-g`** will include debugging symbols, etc.

Where is configure hiding?

You're getting your code from CVS, aren't you? Read the Becoming a GNOMEr section below. If you got a tarball, configure should be in the top directory of the source. If it's not there, that's a bug. See the How do I report a bug? question.

OK, I've got a Makefile now. How do I get GNOME on my system?

From here, it's pretty straightforward. You type **make**, followed by **make install**. On most systems (including GNU/Linux), this should be followed by **ldconfig**. If you're in a rush, you can sometimes skip **make**, since **make install** will usually make everything that hasn't been made yet. If you want to start the whole process and forget about it, and are running the Bourne shell or its derivatives (eg. ksh or bash), put all four commands on one line, separated by **&&**. For example:

```
$ ./configure -prefix=/usr/local && make && make install && ldconfig
```

The **&&** makes sure that the first command succeeded before going onto the next one.

Once you've done **make install** (and **ldconfig** if needed), you can continue on to the next package.

How do I put GNOME someplace special, like `/opt/gnome`?

Many people like to put GNOME off by itself, in its own directory. Doing this makes it easier to uninstall in an emergency, easier to control who has access to GNOME, it lets you export the gnome directory to other systems, and so on. To do this requires a little bit of preparation, and knowledge of your system. For this discussion, I'll be putting GNOME in the `/opt/gnome` directory. Just replace `/opt/gnome` with whatever other directory you want.

The biggest issue are making sure the system can find the GNOME binaries in `/opt/gnome/bin`, the libraries in `/opt/gnome/lib`, and the man files in `/opt/gnome/man`.

To tell the system to look into `/opt/gnome/bin` to find the GNOME binaries, just make sure it is included in the PATH environment variable. For example:

`PATH="/bin:/usr/bin:/opt/gnome/bin"`.

Then, you generally need to tell the loader where to find the libraries. On most commercial Unixes, you add `/opt/gnome/lib` to the environment variable LD_LIBRARY_PATH, in the same manner as PATH, above. On some systems, you use LD_RUN_PATH instead. On systems using ELF format binaries, such as GNU/Linux or FreeBSD, you add the library path to the file `/etc/ld.so.conf`, and run **ldconfig**. On statically linked systems, you don't have to worry about any of the above, but buy more hard drive space, you are sure to need it. Consult your system documentation if you are unsure how to tell the loader which directories to search.

You also should tell man where to find man pages installed by GNOME. Simply insert a line into `/etc/man.config`, saying **`MANPATH /opt/gnome/man`**.

The other issue is there are a couple of directories which GNOME will put things in, which would be better put elsewhere. The easiest way of doing this is with symbolic links. Type the following:

```
$ mkdir /opt/gnome/share
$ cd /opt/gnome/share
$ ln -s /usr/share/locale locale
$ ln -s /usr/share/aclocal aclocal
```

Now you're all set to use `/opt/gnome` as a prefix when compiling. To do this, just give the **./configure** command the option `-prefix=/opt/gnome`. Of course, if you want someplace special other than `/opt/gnome`, just replace it with the directory you do want in all the instances above.

How do I compile GNOME with all the files in the right place for FHS Compliance? How do I relocate GNOME's etc and var directories to easily network GNOME?

* *Thank you Raja R. Harinath!*

By default, GNOME puts all of its files under its prefix directory. While this makes GNOME easy to isolate, it's not necessarily the best place for certain files in some situations. In particular, if you are looking to network GNOME, or for strict FHS (http://www.pathname.com/fhs/2.0/fhs-toc.html) compliance, you might want to relocate some directories. To relocate GNOME's etc directory, use the `-sysconfdir` option. To relocate the var directory, use the `-localstatedir` option.

For example, an FHS installation of GNOME might have a prefix of `/opt/gnome`. However, it's etc files should be in `/etc/opt/gnome`, and similarly, its var files should be in `/var/opt/gnome`. This has the

advantage of allowing you to network the `/opt/gnome` directory, keep it read-only, and allow machines to have local configuration and information without worrying about other machines. To do this, the configure command would be as follows: **`./configure -prefix=/opt/gnome -sysconfdir=/etc/opt/gnome -localstatedir=/var/opt/gnome`**.

I'm done with this module, how do I get rid of all the object files and other cruft from the source tree?

You have three options, **make clean**, **make distclean** or removing the directory. Using **make clean** will remove all the object files, built executables, and miscellaneous cruft, it's nice and safe. The next option, **make distclean** should remove everything that wasn't in the original tarball. If you do this, you will not be able to uninstall the module. The same goes for removing the source directory. Basically, **make clean** is good for general use, the others should be considered drastic measures.

How do I uninstall a module?

If you are using a package management system (such as RPM), just follow its documentation for uninstalling.

If you have installed a module from source code, and you think you may want to uninstall it, *make sure you keep your source tree*. Just go to the module's source and run **make uninstall**. It should go through and remove all the files it installed.

Chapter 4. So now I’ve got GNOME, what do I do with it?

Now we talk about getting started with GNOME, and things that you can do with it.

Where should I look first?

There is a slightly outdated but still useful tour to GNOME at http://www.gnome.org/start/gnometour.shtml. It is certainly worth a look.

Also, make sure you check out the GNOME Users Guide at http://www.gnome.org/users-guide.

How do I start GNOME?

The proper way to start GNOME with all its features is to put the line **`exec gnome-session`** at the end of your X startup script. This script can be called `~/.xinitrc`, `.xsession` or `.Xclients`, depending on your platform and configuration. Of course, make sure that the GNOME binaries are in your path, and there are no lines starting with **`exec`** prior to the one you just added. With gnome-core versions 1.04 and later, you can select the window manager run by setting the WINDOW_MANAGER variable to the one you want.

Running **gnome-session** like this will run the default session, the first time in. This includes your window manager of choice (or a good guess if you didn’t specify it), the panel, GNOME Midnight Commander, and the help browser. As you use GNOME, session management will save your current desktop. If you log out, and log back in later, your most recent desktop will be restored.

If you don’t want session management, GNOME will work fine without it. Put the line **`exec gnome-wm`** at the end of your X startup script, as described above. It will still use the WINDOW_MANAGER environment variable to select your window manager, but not bother with session management. Run panel, gmc, or whatever as desired, either manually, or in the startup script before the exec line. For example:

```
panel &
gmc &
exec gnome-wm
```

How do I run GNOME programs?

First of all, GNOME programs can be run just like any other programs. You go to a command prompt, you type their name, and the program runs.

Secondly, you can run programs from the panel menu. Most GNOME programs (and some non-GNOME ones too) are installed on the Foot menu in the GNOME panel. Some programs of note on this menu; File Manager runs the GNOME version of Midnight Commander, Settings->Menu Editor will let you add or remove things from the Foot (or other menus), Settings->GNOME Control Center will get you the program to configure most of GNOME (and a few other things like Enlightenment), Games->Gnobots is a very addictive game.

How do I run the GNOME Panel?

The GNOME panel is often started from **gnome-session**, if it is not, you can just type **panel** at any command prompt to start the panel.

Add things to the panel by either the Foot's Panel menu, or by the menu you get by right clicking on the panel. The GNOME User's Guide (http://www.gnome.org/users-guide) has an entire Chapter (http://www.gnome.org/users-guide/paneluse.html) on how to use the Panel.

How can I use WindowMaker with GNOME

In order to get the gmc and panel menu stuff on the root window under Window Maker, you need to move the Window Maker Applications menu so that it is activated by the middle or left mouse button. By default, the gmc and panel root menu is mapped to the right mouse button. Unfortunately, the Window Maker Applications menu is also mapped to the right mouse button by default.

To correct this, bring up the Window Maker Configuration tool by going to the Control Panel, selecting the Window Manager subsection under Desktop and hitting the "Run Configuration Tool For Window Maker" button. When the configuration tool comes up, select the mouse icon. Under the "Workspace Mouse Actions", move the Applications menu to the middle mouse button, move the "Select windows" to the left mouse button and deselect the "Window list menu" option (you can use the GNOME pager to do the same thing as the Window list menu does). Save the new configuration. The next time that you use the right mouse button the root menu, you should get the gmc and panel menu. Under there, you can rescan/refresh the desktop and recreate icons, etc.

Chapter 5. Help!!!!!!!!

Even if you have followed every instruction to the letter, sometimes unforeseen problems arise. Anything can go wrong, from a misreading of the instructions, to a bug in the software, to a bug in someone else's software. Here, we try to address problems that people have run into with GNOME.

General Help.

Here are some more general questions that arise often. Specific problems are later in this chapter.

Something is not working, I'm confused, what do I do?

OK, first thing you do is pause, relax, and take a deep breath. Next, look through this FAQ, the Users Guide, and any other documentation you have handy and see if any of it comes close to solving your problem. Next, look through the mailing list archives (instructions are given above) to see if someone else talks about your problem. Then try whatever it was you were doing *one more time* to see if the solution suddenly presents itself. If none of that works, it's time to ask on the list.

First of all, you are subscribed to gnome-list, right? If you aren't, instructions on how to subscribe are given above. Now, when asking for help on GNOME-list, *give lots of detail*. Posting "The panel doesn't work, this sucks!" doesn't help you or us. You still have the problem, and we have a possible bug which we have no information about. It is more useful (but still not sufficient) to say "Whenever I select the foo item from the Foot menu, the entire panel disappears suddenly; by the way, what is this file that GNOME keeps putting in my home directory, it's called core."

Ideally, we need several pieces of information from you. First, exactly what kind of system are you running on? No, not everyone runs the same operating system you are using. We need processor type: Sparc, MIPS, i386 (this includes Pentiums, AMD, Cyrix, etc), PPC, Alpha, ARM, 680x0, et al; we need operating system: Linux distribution, FreeBSD, HURD, OpenBSD, Solaris, IRIX, CrayOS, OS/400, whatever. Also include the version of the operating system. For example, the machine I'm on right now is a GNU/Linux RedHat/i386 Rawhide machine, a friend of mine uses a Solaris Sparc 2.6 machine.

Next, we need some relevant vital statistics of what is on your system. On a GNU/Linux system, the kernel and libc version numbers are almost always relevant. On almost all GNOME problems we need your glib, gtk+, Imlib, ORBit and gnome-libs versions. If you aren't sure if you need any of the above, include them anyway. Other things that are sometimes relevant are versions of Perl, X11, Python, Guile and libpng, unless you see an error message that mentions one of them, you can leave them out. Also important is how you installed GNOME. Did you use binaries, if so which ones? Did you use tarballs? Did you use CVS, if so which dates and/or tags? If you used a mixture of the above, tell us.

Lastly, we need the problem itself. We need to know exactly what you did to get the problem. We need to know if the problem is consistent or intermittent. We need to know if the program output anything prior to the problem. We need to know exactly what happens when you run into the problem. If you want to get ambitious, try some of the diagnostic tools I discuss below, such as a backtrace, and include the information in the mail.

Now before you are ready to post, you need to come up with a good subject line. Gnome-list can have a lot of messages on it, you want yours to stand out to the people you need help from, and be safely passed over by people who can't help. A subject of "GNOME is broken" is likely to get passed over; one of "GSM sucks" will get people in an angry mood, not a helpful one; something like "Most panel icons on HP-UX are just black squares" is likely to get the right sort of attention.

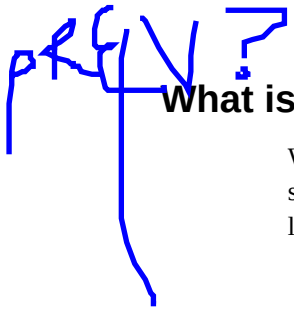
Put this all together into one long email and post it onto the list. If you followed my advice, you are more likely to get a quick, thoughtful response. If for some reason you don't get a quick response, don't take it personally, the mail probably just got lost in the volume. Wait a couple of days, see if there is any detail you can add, or perhaps a more descriptive header, and post it again.

How do I report a bug?

If you find a bug, we want to know about it, we want to track it, and we want to fix it. To make this process as efficient as possible, we borrowed Debian's excellent bug tracking system. Full instructions on how to use it can be found at http://bugs.gnome.org/Reporting.html (http://bugs.gnome.org/Reporting.html).

There are a few things to keep in mind in sending bugs to the system. First make sure to check the web front end at http://bugs.gnome.org to make sure that your bug isn't yet known to the system. If it is a new bug, send it to submit@bugs.gnome.org. Make sure you include the Package and Version headings as described in the instructions, this will make sure it gets routed quickly and automatically to the right people. Make sure you include full details. See the previous question for some suggestions on what is useful to include. If a backtrace or strace is appropriate, by all means include it. Instructions on how to do this is given below.

After you submit the bug to the system. You will get feedback on the bug via email. You also can track the status of the bug through the bug tracking system's web front end.



What is a prefix (or a $prefix, ${PREFIX}, or <prefix>)?

When GNOME is compiled, you need to specify where it is located in the filesystem. The option used to specify this is called `-prefix`, so we call it the GNOME Prefix, or prefix for short. GNOME's file locations should be the same from system to system, except for the prefix, so a file in

`/usr/share/gnome/pixmaps` on one system may be in `/opt/gnome/share/gnome/pixmaps` on another, and we would refer to the location as `$prefix/share/gnome/pixmaps`.

If you installed from tarballs (or CVS), but never gave **configure** (or **autogen.sh**) a prefix, the default is `/usr/local`. If you installed the RedHat RPMS, they use a prefix of `/usr`. If you still aren't sure, type **which panel**, and remove the `/bin/panel` from the end. What's left is your prefix.

What does `Segmentation Fault` mean?

Most Unix and Unix like systems have a feature called memory protection. What this means is the program is allowed to access some parts of memory (often called segments), but not others. If the program tries to access memory outside of the segments it is permitted to access, the system detects the problem and signals an error. This error is called a Segmentation Fault (abbreviated "segv" on some systems), and generally, the program will end suddenly, often leaving behind a core dump (see below).

What it all boils down to is there is a bug in whatever program you were running. The bug might be in the GNOME source code, it might be in the shared libraries your program uses, or it might be in the way the program was compiled. If you are unsure whether it is a bug in GNOME or somewhere else, ask about it on the list. If you are sure the bug lies within GNOME, please make sure it's on the bug tracking system.

What is a core dump? What is this file named core that I keep running into?

In the early days of electronic computers, computer memory was a bunch of magnetic rings called cores, mounted on wires. This type of memory was called core memory. When a programmer wanted to see what was going on in their program, they would output a copy of what was in their core memory so they could analyze it. This output was called a core dump, and the term has remained long after core memory became obsolete.

In many Unix and Unix-like systems, the system will automatically produce a core dump when a program fails catastrophically, for example when it has a Segmentation Fault (see above). It generally puts the core dump into a file named `core` in the directory you were in when you ran the program. This file is very useful for debugging why the program failed, and can be used to produce a backtrace (see below). On the other hand, this file can be quite large, particularly if you aren't using it for debugging. If you can't afford to have core dumps, consult your system documentation to find out how to disable this feature.

What is a backtrace? Someone just asked me to send them a

backtrace, how do I do this?

An incredibly useful debugging tool is the backtrace. This is essentially a snapshot of exactly where in the program you are. It is most often done when a program crashes (exits unexpectedly, often with a segmentation fault or other error) or hangs (stops working but doesn't exit). In this answer, I'm going to have to assume you are using the GNU project's GDB debugger, since that's the most common one used with GNOME, and the one I've used. If you use some other debugger, consult your documentation.

There are two ways to do a backtrace. You can use a core file left when the program crashed, or you can run the program from within the debugger. I have found that running the program within the debugger is more reliable, but I will discuss using core files as well.

To run a program within GDB, go to an xterm (or equivalent), and type **`gdb <full name and location of program>`**, for example, to debug gtalk, you might do **`gdb /usr/local/bin/gtalk`**. GDB does not search the path, so you have to specify where the program is. If you want to run the program with arguments, *don't put them here*. You can specify command like arguments later.

GDB will then identify itself, and give you a `(gdb)` prompt. From here, you type **`run`** to start the program. If you want to run the program with arguments, put them here. For example, to debug gtalk with sound disabled, you would type **`run -disable-sound`**. The program will then start. Keep in mind the program will be slower than normal, and it will use a lot more memory.

Now you try to recreate the problem you had. Do whatever it is that caused the problem before. Once the program crashes, you should immediately see stuff in the debug session, and a `(gdb)` prompt. If the program hangs, wait until you are sure it is hung in the right place, and then hit Control-C in the debug session. It too should reply with a bunch of stuff, followed by a `(gdb)` prompt. At this prompt (whichever way you got to it), type **`bt`**. It will then reply with the actual backtrace.

Here's an example of a debug session. If you are asked for a backtrace, you should send in the whole thing:

```
$ gdb /usr/local/bin/gtalk
GNU gdb 4.17
Copyright 1998 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux"...
(gdb) run -disable-sound
Starting program: /usr/local/bin/gtalk -disable-sound

Program received signal SIGINT, Interrupt.
0x4049290d in __poll (fds=0x808c110, nfds=3, timeout=-1)
```

```
    at ../sysdeps/unix/sysv/linux/poll.c:45
../sysdeps/unix/sysv/linux/poll.c:45: No such file or directory.
(gdb) bt
#0  0x4049290d in __poll (fds=0x808c110, nfds=3, timeout=-1)
    at ../sysdeps/unix/sysv/linux/poll.c:45
#1  0x403c54e9 in g_main_poll (timeout=-1, use_priority=0, priority=0)
    at gmain.c:991
#2  0x403c516e in g_main_iterate (block=1, dispatch=1) at gmain.c:789
#3  0x403c58a1 in g_main_run (loop=0x80882e8) at gmain.c:912
#4  0x401cbf1d in gtk_main () at gtkmain.c:475
#5  0x804a629 in main (argc=2, argv=0xbffffaa4) at main.c:54
#6  0x40407c77 in __libc_start_main (main=0x804a570 <main>, argc=2,
    argv=0xbffffaa4, init=0x8049e10 <_init>, fini=0x804c7dc <_fini>,
    rtld_fini=0x40009c10 <_dl_fini>, stack_end=0xbffffa9c)
    at ../sysdeps/generic/libc-start.c:78
(gdb)
```

To do a backtrace with a core dump is very similar. Change directories to the one which has the core dump, and then run **`gdb -c core <full name and location of program>`**. Again, GDB doesn't look at your path, you have to specify where to find the program. All you then have to do is type **`bt`** at the prompt, and send in the whole session.

What is an strace? Someone asked me to send them one, how do I do this?

The **strace** command performs a "Signal Trace". Using it will allow you to run a program while outputting system calls and signals. This is a much more verbose debugging tool than the backtrace, but sometimes you need the extra information an strace can give you. Getting an strace is easy, just type **`strace -o <output file> <command with args>`** into an xterm. For example, to do an strace on the gtalk program with sound disabled, I might do **`strace -o /tmp/gtalk.strace gtalk -disable-sound`**. This will save the strace output in `/tmp/gtalk.strace`. Strace does make use of the path, and does accept program arguments on the command line.

Warning, strace produces lots of output (the above strace example came up with 5,498 lines in five seconds). Don't send straces to the mailing lists, and only send them to the developers if they specifically request it.

On Solaris systems, the **strace** command does something different. The **truss** command is probably what the person who asked for the strace is looking for.

Specific Compile Problems

What does `undefined reference to 'compress'` mean?

In general, it means that it is having trouble finding or linking to the compression library (libz). First, make sure it is installed. If it isn't, then find it and install it. It should be included in all major Unixes and Unix-like systems.

If it is installed properly, and you still get the error, than check to see if there are any other copies of libz floating around. One place where it often is hiding when people have problems is in `/usr/X11R6/lib`, since XFree86 puts a copy there. It is probably safe to just delete it, but if you aren't sure, rename it to something like `libz.so.renamed`, that way it won't be found by the linker, and if you run into problems elsewhere because of the renaming, you can just rename it back.

What is `AC_TRY_RUN`, and why is my machine warning me about it?

Sometimes, when running aclocal, autogen.sh or make, you will get lines looking like: `configure.in:171: warning: AC_TRY_RUN called without default to allow cross compiling`. This is nothing to be alarmed about, you can safely ignore it (unless you are cross compiling).

If you're still curious, a cross compiler is a compiler based on one platform that you can use to create code to run on a different platform. For example, some people have a version of GCC installed on GNU/Linux boxes with Intel processors which makes binaries for SGI boxes with MIPS processors. This would be a cross compiler.

The AC_TRY_RUN macro is called to attempt to compile and run a code fragment. It is used to test to see whether a particular piece of code is likely to work on your machine or not, and adjust the configuration accordingly. If you are cross-compiling, however, you can't run the code fragment, since it's on a different machine than the one you are on. The AC_TRY_RUN macro includes a parameter to tell it what to do in this case, but we haven't tested cross-compiling GNOME yet, so it would be premature to configure this value. Because the value is absent, it gives the warning.

While compiling the program `gnome-gen-mimedb` segfaults

This means you have a broken db 1.85 package, please upgrade your db library

Runtime Problems

How do I reset my session management settings?

Many people find session management very useful. However, sometimes session management can go astray, and do undesired things like remember your settings after your kid brother attacked your computer. If GNOME is properly installed, returning your settings to the default is easy.

First, exit completely out of X. If you are using session management, you should be able to use the Log out item of the Foot menu to exit. If that doesn't work, hold down the control and alt keys, and hit backspace. If that still doesn't work, and you are now looking at an XDM or GDM login prompt, switch to a text login (control-alt-F1 on many systems, consult your system documentation if you aren't sure, log in as a different user and use **su** in an xterm if you are really in a bind).

Then, to remove your settings, while you are out of X, remove the `~/.gnome/session` file. The next time you start X, or the next time you log in from GDM, it should start you with the default window manager, panel, gmc and the help browser.

I started GNOME with session management, why am I just getting this grey screen (no panel, nothing)?

If you use session management, and it can't find a session, this is what it gets. First, have you tried resetting your session management settings (See above)? If this doesn't fix it, than **gnome-session** is having trouble finding the `default.session` file. In gnome-core versions 1.03 and later the file should be in `$prefix/share/gnome`. Prior to that it should be in `$prefix/share`. If the file is missing, you should probably reinstall gnome-core.

How do I tell session management what window manager to use?

This is a tougher question than it sounds, since the answer depends on which version of GNOME you are running. The easy answer is to use the Window Manager settings in **control-center**. If that doesn't work for you for some reason, here's how to do it.

In gnome-core version 1.03 and earlier, the window manager is selected by placing it in a file called `default.wm`. The system default for this file should be in either the `$prefix/share/gnome` or `$prefix/share` directories, whichever one has the file `default.session`.

A user-specific version of `default.wm` can be put in the `~/.gnome` directory. The `default.wm` file should look like:

```
[Default]
WM=foo
```

Where **`foo`** should be replaced by the window manager you want. If you want to change the system default, you might also have to change the `default.session` file.

In gnome-core versions 1.04 and later, it is much easier. Set the environment variable WINDOW_MANAGER to the window manager you want, then start X. If that variable is not set, GNOME will run a script to make a good guess as to your window manager. You can also edit the `~/.gnome/default.wm` file as in older versions.

Everything seems to be working, but most of my icons are just black squares, or random spots. How do I get my icons?

Most of GNOME's icons are PNG (http://www.cdrom.com/pub/png) format graphic images, displayed via libpng and the Imlib graphics library. The trouble is, this part of Imlib seems to be very finicky, and it shows its displeasure by displaying garbage (the black squares are just cleaner garbage).

There are a few things that can cause this to happen. The most common problem is a bad, missing or confused libpng. If you have libpng version 1.0.2, assume it is bad. Only use versions 1.0.1, 1.0.3 or later. Missing is easy, if there is no file called "libpng.so" or "libpng.so.2", and you are on a system with dynamic libraries, it's missing, get a copy and install it. To see if it's confused, see if you have more than one copy of "libpng.so" or "libpng.so.2", if you do, remove them all and reinstall the right one.

Another possibility is one I have on one of my machines (A RedHat box with an AMD5x86 processor and egcs). If I use any optimization when I compile glib or imlib, I get the problem described (imlib uses the gmodule portion of glib to load the PNG code). Try compiling those two packages with no optimization switches on. That might fix it. As far as I can tell, this is a bug in my compiler, not in GNOME.

I have also received a report that the problem could occur from interference from some versions of ImageMagick. If you are at a loss, you might want to try removing or upgrading it.

Chapter 6. Architecture Components

This section discusses the various pieces of GNOME, and what they do.

Libraries

Why do I need so many libraries for GNOME?

GNOME requires the libraries it does for two big reasons. We wanted GNOME to have lots of useful features, and we didn't want to write everything from scratch. By using common libraries for many of GNOME's features it allows GNOME to work better and more consistently with other non-GNOME applications.

CORBA, ORBit and Bonobo

Network Objects is GNOME's middle name. This is how they work.

What is CORBA?

CORBA (http://www.omg.org/news/begin.htm) is the Common Object Request Broker Architecture. From the Open Management Group (OMG, the standards body which controls CORBA) document What is CORBA? (http://www.omg.org/about/wicorba.htm) comes the following description:

> The Common Object Request Broker Architecture (CORBA), is the Object Management Group's answer to the need for inter-operability among the rapidly proliferating number of hardware and software products available today. Simply stated, CORBA allows applications to communicate with one another no matter where they are located or who has designed them(...)
>
> The (ORB) is the middleware that establishes the client-server relationships between objects. Using an ORB, a client can transparently invoke a method on a server object, which can be on the same machine or across a network. The ORB intercepts the call and is responsible for finding an object that can implement the request, pass it the parameters, invoke its method, and return the results. The client does not have to be aware of where the object is located, its programming language, its operating system, or any other system aspects that are not part of an object's interface. In so doing, the ORB provides inter-operability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems.

In fielding typical client/server applications, developers use their own design or a recognized standard to define the protocol to be used between the devices. Protocol definition depends on the implementation language, network transport and a dozen other factors. ORBs simplify this process. With an ORB, the protocol is defined through the application interfaces via a single implementation language-independent specification, the IDL (Ed. Interface Description Language). And ORBs provide flexibility. They let programmers choose the most appropriate operating system, execution environment and even programming language to use for each component of a system under construction. More importantly, they allow the integration of existing components. In an ORB-based solution, developers simply model the legacy component using the same IDL they use for creating new objects, then write "wrapper" code that translates between the standardized bus and the legacy interfaces.

OK, enough of that stuff. Here's Todd's explanation:

Do you remember RPC? You know, Remote Procedure Calls? Sun used them as the basis for NFS and NIS. Microsoft used the Distributed Computing Environment's RPC standard as the basis for DCOM.

Well, a remote procedure call is really pretty simple. First, you define the procedure in some standard format. What is there to a procedure definition? Well, you've got the name of the procedure, the arguments which go into it, and the results which come out of it. Then you produce the client side, which tells clients how to feed in arguments and get back results, and then you build the server side, which takes in arguments and gives back results.

You can use this model to build pretty powerful systems. Just take the standard Unix file system interface, convert it to RPC, and bammo, you've got NFS. Note that as long as the inputs and outputs are thrown around in a standard way, the clients calling the procedure and the server servicing the procedure don't have to be written in the same language, running on the same machine, or even running on the same operating system or hardware. This is, truth be told, the killer feature of RPC.

However, as procedural programming (C, Perl) has been superseded by object-oriented programming (Objective C, Java, C++), you really need something more the procedures in RPC. You need something which supports objects: creating objects, accessing object data, accessing object methods, destroying objects. This is where CORBA comes in: think of CORBA as next-generation RPC, just extended to support object-oriented programming. Instead of having under RPC the following:

```
void foo(int bar); void baz(){return(-1);}
```

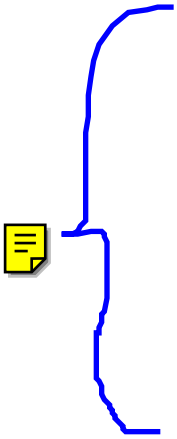
you instead have under CORBA the following:

```
interface bubba{ void foo(in int bar); void baz() raises (InValidContext); }
```

The lesson of all of this? CORBA lets one program language-independent, location-transparent object-oriented interfaces between software components. It' s cool.

What are some CORBA components?

I would like to include here definitions of some of the more strange CORBA terminology, so that people can easily get into the groove of what CORBA is and have a good "back-of-the-envelope" understanding of the components involved. Whenever I refer to a non-obvious CORBA term, I intend to define it here.



- *Object Request Broker* - (or ORB) an ORB is a piece of "middleware", as it's called, which sits between clients and servers and makes easy communication between them possible. The ORB is a conceptual entity, which sometimes takes the form of a shared library and sometimes takes the form of an external program. the ORB is responsible for establishing and destroying sessions between clients and servers and marshaling, unmarshaling, transporting messages between them during a session.
- *Object Adaptor* - (or OA) an Object Adaptor provides the channel through which an object server (like the gnome panel) communicates with the Object Request Broker (ORBit).
- *GIOP/IIOP* - CORBA takes all of these object-oriented language-isms and handles carrying them between object-oriented software components (marshaling, transporting, and demarshaling). If the two pieces of software are in different locations, then it will use the GIOP protocol to move information between them. IIOP is a specification of GIOP for the Internet protocol suite; theoretically you could run GIOP over something other than IP, in which case it would be called something else. I for one can't figure out why you would want to. If any version of GIOP other than IIOP ever became important, then I'm sure that ORBit would support it, too. (Think of IIOP as HTML to GIOP's SGML.)

What are CORBA Services?

Additionally, there are really two parts to CORBA. There is the part which I've already described, which tells you how to do objectified RPC, how to marshal arguments, how to write IDL files, how GIOP works, etc. and then there are the "CORBA Services". These are services built on top of the basic CORBA infrastructure which make distributed object programming easier.

Perhaps the most important of these is the naming service. E.g., say that you were a program which for some reason needed spell-checking services. One really neat thing that you can do (if your CORBA implementation is nifty enough) is to call up your ORB and say "Hey! I need spell-check service. Go find it for me." The ORB then uses the naming service to see what objects are registered which provide spell-check service. It can then wake up the calling program and say "You can find the spell- check service at #867-5309." You then call up 867-5309 and, lo and behold, there's a spell check service there. Now, that might be a shared library which got mapped into your address space and you are now doing straight function calls into it, or it might be a community spell-check service in Mongolia offered as a public service by the Mongolian FreeBSD users' group over the internet via IIOP. You don't know and, except for maybe a little lag induced by the round-trip to Mongolia, you really don't care. Cool, non?

The naming service, then, allows objects to register themselves for future use. There are other CORBA services, including a transaction service, a security service, a time service, a "relationship service",

which might or might not be what you think it is, and more. My copy of the CORBAservices document from March 1997 lists 14 of them.

You can find a copy of the CORBAservices specification at the OMG web page at http://www.omg.org/corba/csindx.htm. If you are interested in implementing any of them, then I'm sure that the ORBit developers would love to talk to you. (See ORBit)

Which CORBA version does GNOME use?

Work is presently underway within the OMG to develop version 3.0, but for now GNOME is using CORBA version 2.2, which is the present standard. I have not heard what GNOME will do when 3.0 comes out.

Where can I find out more about CORBA?

A good jumping-off page is Linas Vepstas's CORBA page at http://linas.org/linux/corba.html (http://linas.org/linux/corba.html).

Since there seems to be a lot of interest in the difference between CORBA and COM, I'll just mention this one other paper: http://www.research.att.com/~ymwang/papers/HTML/DCOMnCORBA/S.html, DCOM and CORBA Side by Side, Step by Step, and Layer by Layer.

Finally, the TAO project at Washington University in St. Louis has their own ORB, TAO. That group is led by Dr. Douglas Schmidt. You can find a lot of good information on his CORBA page: http://siesta.cs.wustl.edu/~schmidt/corba.html. Be sure to look around at the ACE and TAO pages while you're there; they are very good.

What role does CORBA play in GNOME?

CORBA enables the component architecture of GNOME. It serves a role similar to that of COM/DCOM under win32.

What CORBA implementation is GNOME using?

In the beginning, the plan was to use ILU ( //parcftp.parc.xerox.com/pub/ilu/ilu.html). ILU had a number of benefits, foremost among them that it supports several languages. While ILU was (and is) awfully neat, Xerox's attitude towards it was unclear, and the licensing terms proved to be fatal: ILU is

not free software, and the GNOME team proved unable to get Xerox to modify the license. The GNOME project won't adopt technology which is not free software, and that was the end of that.

The project then settled upon MICO (http://diamant-atm.vsb.cs.uni-frankfurt.de/~mico/).  main attractions of MICO are that it has an Object Adaptor, it is IIOP-compliant, and it is licensed under the terms of the GPL. However, it does not presently support languages other than C++, and it used just an unbelievably amount of memory.

Still dissatisfied, the GNOMErs began work on our own ORB, called ORBit.

What is ORBit?

ORBit is intended to be multilingual; ILU proves that this is possible. Right now it only supports C, but in the future it will support other languages. (Really, we mean it! It is a very, very new project right now, which is the only reason it is C only.) It also supports GIOP/IIOP, the OMG's CORBA protocol which allows different ORBs to be able to talk to each other.

Finally, ORBit is intended to be high performance. That means low-memory and high-speed. Here, ORBit is taking a lot of cues from Flick (http://www.cs.utah.edu/projects/flux/flick/). Elliot Lee, the instigator of the ORBit project, thinks that he can, without violating the CORBA spec, get the cost of a CORBA call darn close to that of a standard library call, where a local implementation of the requested service is available. We'll see.

Why has GNOME not made more use of CORBA?

The big reason is that MICO was just unworkable. it took too much memory, it was C++ only, etc. ad naus. With the recent release of the GNOME ORBit, we can start deploying CORBA in GNOME.

How will CORBA be used in GNOME, or, what is Bonobo?

(This section comes pretty much verbatim from Miguel de Icaza, Chief of the GNOMErs.)

CORBA will be used in various contexts. A set of interfaces and library routines called "Bonobo" are used to simplify and integrate applications:

- Components: There are pre-packaged libraries/servers that do provide some functionality that can be used by any application like spelling servers, calendaring services and other non-GUI services.
- Application embedding and in-place activation: Bonobo is a set of interfaces for providing application embedding and in-place activation which are being defined. The Bonobo interfaces and interactions

are modeled after the OLE2 OpenDoc interfaces.

- GNOME Applications as service providers: GNOME applications will export their inner workings through CORBA to allow anyone to reuse their functionality. This allows scripting of the GNOME desktop as any CORBA-aware application or scripting language can invoke methods in a GNOME server (for example, a script in Perl can control remotely invoke a spreadsheet to make a graph or to make a sales report). Some people call this "Automation". Unlike the COM world, the use of CORBA makes this transparent to the implementor of a server.
- Reusable controls: Another set of Bonobo interfaces deal with reusable controls. This is similar to Sun's JavaBeans and Microsoft Active-X.

Bonobo doesn't stand for anything. They are Chimpanzees' less grouchy cousins.

Graphics in GNOME

What is GTK+?

GTK+ is the Gimp Toolkit. It was originally written as part of the Gimp project but it has grown into an excellent general purpose widget set. Find out more at http://www.gtk.org

What role does GTK+ play in GNOME?

GTK+ is the interface which GNOME applications use to interact with the user. It is similar to the role of Motif in CDE, Qt in KDE or the-widget-set-which-has-no-name in Win32.

What languages does GTK+ support?

You can write GTK+ programs in tons of languages! Your humble FAQ author knows of the following bindings:

C (original)
C++
Objective C
Python
Perl 5
Scheme (Guile)
TOM

Gwydion Dylan
Ada95
Pike
Pascal (Free Pascal Compiler)

If there are others, then please let us know about them. (See Contact the FAQ Maintainer)

You are encouraged to add bindings to other languages should the fancy strike you. N.b., that the Scheme bindings already have a formalized representation of the API, so if you want to start working on new bindings, then you might want to start from here.

Having written that, I have to wonder if we can't somehow use CORBA to cut this Gordian knot. If the ORBit developers can get the cost of a local CORBA call down to that of a normal shared-library call, then why not CORBify GTK and work on adding bindings for new languages to ORBit? I know, the conventional wisdom is that CORBA is too heavyweight for high-volume stuff like X calls. Hey, don't blame me for throwing out crazy ideas: the reason I maintain the FAQ is that I can't program!

What is imlib?

Along with GTK, GNOME uses imlib, an image library for X which supports multiple image formats transparently to the programmer. Imlib can even use external converter programs like ImageMagick or pmbplus to convert unknown image formats to known formats, so if a converter exists for an image type, you can almost certainly use that image type in your application! Imlib also supports multiple bit-depths, from 24-bit down to 1-bit monochrome, again transparently to the programmer, and does intelligent color map allocation and automatic dithering to get the best possible performance out of your display. Anyone who has ever tried to do this in their X apps, or who has had to endure their screen flashing when apps have to allocate a private color map, you will especially like this feature.

You can find out more about imlib at http://www.labs.redhat.com/imlib.

What's the deal with themes?

Themes let you change the look and feel of GTK+ applications without recompiling them. With themes, you are able to change the theme of all GTK+ apps on a given display at the same time.

There are actually two parts of a theme in GTK+, the engine and the configuration. The engine is a specially constructed shared library with instructions on the themes way to draw the widgets (the controls of a window, like the buttons, scrollbars and menus). The configuration includes information like fonts, colors, which engine to use, and sometimes graphic images to incorporate into the theme. Together, the two make the Theme, and give GTK+ its flexible appearance.

To see some examples of what themed GTK+ apps look like, take a look at: http://gtk.themes.org (http://gtk.themes.org)

How do I get themes working?

First, you need a recent version of GTK+, I recommend version 1.2.6 at the moment, but 1.2.0 works fine as well. You then get the themes you want. A good selection of themes is available in the gtk-engines module. Even more are available at http://gtk.themes.org (http://gtk.themes.org).

Many of the themes include an engine. You need to compile the engine, and install it. In gtk-engines, this is done the same way as any GNOME module. The themes from the website might have different instructions, check their documentation. Some of the themes on the website are merely alternate configurations for another engine, many of them are for the pixmap engine in gtk-engines.

Once the theme is installed, you can select them through the GNOME Control Center, under Desktop->Theme Selector. The selector will allow you to choose between all the themes installed on the fly, preview what the the theme will look like, and in version 1.04 or later even let you select the font.

* *You can see what the FAQ author's themefied GTK apps look like at: http://www.mindspring.com/~tlewis/screen.jpg (http://www.mindspring.com/~tlewis/screen.jpg). Commented out due to dead link.*

What is OpenGL?

OpenGL is a 2D and 3D rendering API developed by Silicon Graphics but has become a de facto industry standard. Basically, it is a tool to take a viewpoint of 3D space, and produce a 2D image of it suitable for drawing on a screen. To quote SGI's FAQ:

> OpenGL is a rendering only, vendor neutral API providing 2D and 3D graphics functions, including modeling, transformations, color, lighting, smooth shading, as well as advanced features like texture mapping, NURBS, fog, alpha blending and motion blur. OpenGL works in both immediate and retained (display list) graphics modes.

GNOME uses an implementation of the OpenGL API called Mesa. Mesa is not OpenGL per se, since it is a Free Software project that has not gone through the OpenGL trademark licensing process, but it is a faithful representation of the OpenGL API. Mesa is not OpenGL(tm) in precisely the same way that GNU/Linux and FreeBSD are not Unix(tm).

What about Drag and Drop support?

GNOME currently uses the Xde Drag and Drop protocol. XDE is being replaced with an implementation of the X Windows Drag-And-Drop Protocol to inter-operate with other toolkits. Find out more at http://www.cco.caltech.edu/~jafl/xdnd/ (http://www.cco.caltech.edu/~jafl/xdnd/).

Session Management

What is session management?

Session Management is an extension to the X Window system which allows applications to save and restore their state. This allows users to have their desktop look like they left it, rather than having a clean desktop every time they re-enter the GNOMosphere.

How do I use session management in my GNOME application?

Dr. Mike of RedHat has written a tutorial on this: http://www.gnome.org/devel/start/sm.shtml (http://www.gnome.org/devel/start/sm.shtml).

DocBook

What is DocBook?

DocBook is a SGML DTD (a set of tags for a text file) that is very useful for writing documentation. Most of the GNOME documentation is written in DocBook. To find out more, look at Oasis's DocBook pages at http://www.oasis-open.org/docbook. There is also a good tutorial for DocBook at http://nis-www.lanl.gov/~rosalia/mydocs/docbook-intro.html.

The version of DocBook, and related tools, as used by GNOME can be found at ftp://ftp.cygnus.com/pub/home/rosalia/docware. Tarballs and binary RPMs are both available there.

What is SGML?

To be filled in.

Internationalization & Localization (I18N & L10N)

What are Internationalization(I18N) and Localization(L10N)?

From the GNU gettext info page:

By *internationalization*, one refers to the operation by which a program, or a set of programs turned into a package, is made aware and able to support multiple languages. This is a generalization process, by which the programs are untied from using only English strings or other English specific habits, and connected to generic ways of doing the same, instead. Program developers may use various techniques to internationalize their programs, some of them have been standardized. GNU gettext offers one of these standards.

By *localization*, one means the operation by which, in a set of programs already internationalized, one gives the program all needed information so that it can bend itself to handle its input and output in a fashion which is correct for some native language and cultural habits. This is a particularisation process, by which generic methods already implemented in an internationalized program are used in specific ways. The programming environment puts several functions to the programmers disposal which allow this runtime configuration. The formal description of specific set of cultural habits for some country, together with all associated translations targeted to the same native language, is called the locale for this language or country. Users achieve localization of programs by setting proper values to special environment variables, prior to executing those programs, identifying which locale should be used.

What I8N and L10N API does GNOME use?

GNOME uses the Uniforum’s internationalization standard. This is the "standard" invented by Sun Microsystems and adopted by the GNU project in their "gettext" package. N.b., this is different from the competing X/Open standard.

How do I program using I18N and L10N, or Where do I find out more info on these topics?

Take a look at the GNU gettext package’s online manual at http://www.gnu.org/manual/gettext/html_chapter/gettext_toc.html. There also is a very useful article from Linux Journal about internationalization issues at http://www.linuxjournal.com/issue59/3286.html (http://www.linuxjournal.com/issue59/3286.html). Additionally, you may find the I18N FAQ informative: http://www.vlsivie.tuwien.ac.at/mike/i18n.html.

How do I change the language of my GNOME programs?

To change the language that GNOME programs use, you just need to set the environment variable LANG to the desired ISO 639 language code. From that point forward, any new GNOME apps you run will try use that language.

Let's say you wanted to see GNOME in Spanish. In the Bourne Shell and its derivatives (such as Bash), you use **set LANG="es"**. If you were using the C Shell (or tcsh), you would use **setenv LANG "es"** instead. These commands are generally put in the `.profile` or `.login` file, so that your environment starts up the same way each time.

Some languages (such as English and Spanish) need to be handled differently depending on where you are. In these cases you can extend the language code with an underscore followed by a ISO 3166 country code. So, if you want to select British English you would use "en_GB"; for Mexican Spanish you would use "es_MX".

Keep in mind that GNOME apps that are already running won't change languages while you change the LANG variable, you have to close the program and reopen it for the new value to take effect. Also note that some languages are not offered in all applications. You might find some apps using the language you want, and others using English.

You can find the list of all the ISO Language and Country Codes at http://www.twics.com/~craig/writings/linux-nihongo/node93.html. Here's the list of the codes currently supported by GNOME-core:

Catalan (ca)
Czech (cs)
Danish (da)
German (de)
English [USA] (en) [Yeah, life's not fair]
English [Great Britain] (en_GB)
Spanish [Spain] (es)
Spanish [Dominican Republic] (es_DO)
Spanish [Guatemala] (es_GT)
Spanish [Honduras] (es_HN)
Spanish [Mexico] (es_MX)
Spanish [Panama] (es_PA)
Spanish [Peru] (es_PE)
Spanish [Sweden?] (es_SV) [Should be es_SE if its Sweden]
Basque (eu)
Finnish (fi)
French (fr)
Irish (ga)
Hungarian (hu)

Italian (it)
Japanese (ja)
Korean (ko)
Dutch (nl)
Norwegian (no)
Polish (pl)
Portuguese (pt)
Russian (ru)
Slovak (sk)
Swedish (sv)
Waloon (wa)

Guile

What is Guile?

Guile is the GNU project's implementation of Scheme, a Lisp dialect. Find out more at http://www.gnu.org/software/guile/guile.html. There's also some excellent information at http://www.red-bean.com/guile (http://www.red-bean.com/guile).

The Guile docs are pretty good, but the Scheme standard (the wonderfully, obscurely named R5RS) can be found at: http://www-swiss.ai.mit.edu/~jaffer/r5rs_toc.html.

What role does Guile play in GNOME?

Guile is the scripting or "glue" language for the GNOME project. From the original announcement:

> We plan to use GTK/Scheme bindings for coding small utilities and applications. When these bindings are more mature, it should be possible to write complete applications in Scheme.

Chapter 7. Developer Issues

What does a window manager have to do to be GNOME-compliant

There are three big things:

- It must be fully ICCCM compliant, especially with proper X Session Management support.
- It must support MWM hints.
- It must support GNOME hints.

For more detailed information, see the GNOME Window Manager Compliance page at http://www.gnome.org/devel/gnomewm/book1.html (http://www.gnome.org/devel/gnomewm/book1.html)

What is Autoconf? What does it have to do with GNOME?

Autoconf is a tool for producing shell scripts that automatically configure source code to adapt to the system you are using. Autoconf is invaluable, since it makes the development of such configure scripts drastically easier. GNOME uses it alongside the related tool, Automake, which allows Autoconf to produce custom Makefiles for your system. Autoconf uses m4 to operate. M4 is a powerful (but arcane) macro language used by many GNU utilities. Most of the macros used to generate these files are found in either `/usr/share/aclocal` or the `macros` subdirectory of the source tree.

If you are using the tarballs, Autoconf and Automake have already been run for you, so you don't need them on your system. If you use CVS, you won't get very far without having recent versions of these tools. You can find more information about Autoconf at http://www.gnu.org/software/autoconf/autoconf.html (http://www.gnu.org/software/autoconf/autoconf.html), and Automake at http://www.gnu.org/software/automake/automake.html (http://www.gnu.org/software/automake/automake.html).

Keep in mind, that `configure`, `config.h`, `Makefile`, and even `Makefile.in` are all constructed files. If you need to change them, you shouldn't patch them directly, but rather modify the files used to create them: `configure.in`, `config.h.in`, and `Makefile.am`.

What is Libtool? What does it have to do with GNOME?

Libtool is an important portability utility. It gives a common interface to the mechanics of building, debugging and maintaining static and dynamic libraries on most computer systems in use. It works in three phases, the **libtoolize** command adds libtool support to a source tree by adding the **ltconfig** script. This script, in turn, generates a system-specific script called **libtool** in the source tree. The **libtool** script is the part that actually gets all the work done.

Like Autoconf and Automake, if you are using CVS, you need Libtool installed on your system. If you use the tarballs, **libtoolize** as already been run, and **ltconfig** is self-contained, so you do not need Libtool to be installed. More information about libtool can be found at http://www.gnu.org/software/libtool/libtool.html (http://www.gnu.org/software/libtool/libtool.html)

Chapter 8. Becoming a GNOMEr

General information on getting more involved with the GNOME Project.

How do I join the GNOME movement?

If you want to get seriously involved with GNOME, you really have to join the gnome-list mailing list (See Mailing Lists, above). Check the list of mailing lists to see what other lists interest you. Also, make sure you read the GNOME Manifesto at http://www.gnome.org/about/manifesto.shtml and the articles on the GNU Project Philosophy at http://www.gnu.org/philosophy to make sure you know what kind of project you're getting involved in. You don't have to agree with all of the philosophy in order to contribute to GNOME, but at least respect that some of us are committed to the ideas given in these documents.

Since GNOME is a software project, we obviously can always use more programmers, but you don't have to be a programmer to help out with GNOME. We also can always use more people sending in clear bug reports. There is always more documentation to be written. If you speak more than one language, we can always use translators, both of the documentation and of the labels, buttons and menu entries within the applications. People willing to make binary packages for their system are quite welcome, particularly if nobody else is making binaries yet for that system. People willing and able to mirror our WWW, FTP and/or CVS servers are enthusiastically encouraged to do so.

I am a programmer, and I want to help, what can I do?

There are a number of projects being worked on, you might want to check out the `gnome-status` CVS module from the CVS to see a list of projects that people have identified as needing work.

A good rule is that you can always improve the robustness of code, improve the user interface, make the code simpler, make the user interface better, make sure applications are properly internationalized and localized.

How do I get the bleeding edge, CVS versions of GNOME?

If you are developing for GNOME, or if you just want to keep more up-to-date than the latest release, you should consider using our CVS system. There is the main CVS server, which you need an account

for, and a network of mirrors of the server, which anyone can read from. This network of mirrors is collectively called Anonymous CVS, and they all can be accessed at anoncvs.gnome.org (they are set up with round-robin DNS to balance out the load). If you are looking to contribute source code to the GNOME project, you will eventually want an account on the main CVS server, but get used to the Anonymous CVS system first, so you are familiar with how GNOME makes use of CVS.

Directions on how to access the Anonymous CVS system are at http://www.gnome.org/devel/whatiscvs.shtml. There is also a very useful guide at http://www.cinternet.net/~soren/gnome/cvs.shtml. The anonymous CVS servers should be updating themselves

In order to use the CVS version, you need to have GNU tools installed on your system. You need at least GNU Autoconf, Automake, Libtool and m4. If your system doesn't include these, you can get them from the GNU website at http://www.gnu.org. 

I want bleeding edge GNOME, but I can't use CVS. How do I get it?

There are some occasions where CVS just isn't feasible. Perhaps you want to make sure GNOME works using just the native tools for your system instead of the GNU tools. Perhaps you are sitting on the other side of a firewall that blocks the CVS port. Whatever your reason, Jim Pick Software offers daily snapshots of the CVS version, in tarball form, available via FTP at ftp://ftp.jimpick.com/pub/gnome/snap. The conversion to tarball form bypasses the need for GNU tools as well. 

How do I get an account to let me contribute to the CVS version of GNOME?

If you plan on contributing code to the project, then you will need an account on the CVS server. Miguel de Icaza (miguel@nuclecu.unam.mx (mailto:miguel@nuclecu.unam.mx)) determines who gets accounts on the CVS server; if you would like one, then send him a message detailing what code you plan on writing, along with with a crypted password. You can get that by doing the following:

```
perl -e 'print crypt ("YourPassword", "salt");'
```


Where the "salt" string preferably should be a random 2-char string.

Why are directories sometimes missing when I update a module from CVS?

CVS has an excellent feature with an annoying quirk about it. CVS allows a module to include other modules, a feature which GNOME uses to better organize things and to reduce duplication of code. Unfortunately, the **cvs update** command doesn't bother to check to see if any new modules are included off of the main module. If you think there is a missing piece, use **cvs get** instead, it will check for new included modules. Don't worry, as long as the source tree is there, it won't try to do a brand new checkout, it will act the same as **cvs update** as far as only downloading what is new or updated.

I am not a programmer, how can I contribute?

There are many ways to contribute to the GNOME effort if you are a not a programmer. You can help writing documentation (contact the people at docs@gnome.org for this), you can review the existing documentation and fix typos, format, and make the documentation more clear or enhance the examples.

If you are a non-english speaker and you understand english, you can also help with the translation effort. This means translating application textual strings and documentation. To work on this, please subscribe to the gnome-i18n@nuclecu.unam.mx mailing list. To subscribe send a message to majordomo@nuclecu.unam.mx and in the body of the message put "subscribe gnome-i18n". You can view the current status for our language translation effort in the http://www.gnome.org/i18n web page.

If you have a good idea or a great concept on how to improve the user interface experience in GNOME, please subscribe to the gnome-gui-list@gnome.org (mailto:gnome-gui-list-request@gnome.org). The user interface team is lead by Jim Cape who maintains a web page with the suggested improvements at http://www.jcinteractive.com/gnome-ui/

Chapter 9. FAQ Issues

Where is the official copy of this FAQ?

The formal and official copy of this FAQ is kept in DocBook (SGML) format on the CVS server at `cvs.gnome.org:/cvs/gnome`, the file itself is located in the `gnome-libs` module in the file `devel-docs/gnome-faq.sgml`. All other versions are copies, ports, reformatings, etc.

How do I add a question and/or an answer to this FAQ?

The easiest way is to email the FAQ maintainer, at gnomefaq@gnu.org (mailto:gnomefaq@gnu.org). If the suggestion is appropriate, we will edit your information, and incorporate it in at the appropriate place.

If you have CVS access, you can modify the FAQ on your own, but I would prefer if you limit direct modification to quick fixes of glaring errors, rather than major changes or additions.

Chapter 10. Credits

Copyright and Disclaimer

This article is provided as is without any express or implied warranties. While every effort has been taken to ensure the accuracy of the information contained in this article, the author/maintainer/contributors assume(s) no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Trademarks and Naming Credits

Here are some of the less obvious trademarks used in this document.

Linux is a trademark of Linus Torvalds.
Debian is a trademark of Software in the Public Interest
Copyleft and Free Software are not trademarks of anyone, but they are important terms in the Free Software community, an
Open Source is a service mark of either Software in the Public Interest or the Open Source Initiative or Bruce Perens, nobo
Unix, and The X Window System are trademarks of the Open Group.
OpenGL is a trademark of Silicon Graphics, Inc.
All other trademarks are the property of their respective owners.

If there is a trademark here that needs to be listed formally above, by all means let us know so it can be added.