1.1 GNUstep General Information

1.1.1 What is GNUstep?

GNUstep is the Free Software Foundation's effort to implement NeXT Software, Inc.'s (now Apple Computer, Inc.) OpenStep Standard. Also we are building developer and user applications based on this standard which may someday be used to form a complete desktop experience.

1.1.2 What is the OpenStep standard?

OpenStep is an Application Programming Interface (API) for creating applications using the Objective-C language. It was published by NeXT Computer, Inc. in 1994.

OpenStep consists of three parts: the '`Foundation Kit`', a library of non-graphical objects; the '`Application Kit`', a library of objects useful in creating graphical applications; and the '`Display PostScript System`' (DPS), an interface for drawing to the screen using the PostScript graphics language. DPS support is not being persued at this time however.

You can obtain a copy of the OpenStep standard from the GNUstep web site `http://www.gnustep.org` or it's mirror sites.

1.1.3 What platforms does GNUstep run on?

See the list of supported platforms at `machines_toc.html` for information on what machines GNUstep builds on and what the status of the ports is. Probably a few days porting to any other UNIX system where current gcc compilers and gdb debugger work.

1.1.4 Does GNUstep run on Windows?

The primary targets for GNUstep are free UNIX system-based platforms such as GNU/Linux and FreeBSD.

That being said, the base library should run on Windows NT, 98, 2000, and XP with the Cygwin UNIX system-emulation environment from Cygnus (`http://www.cygwin.com/`), or the MinGW environment (`http://www.mingw.org`).

The GUI library uses the win32 backend library to work under Windows. The backend library is a thin layer that converts the GNUstep methods to handle drawing of GUI elements to calls to the Windows API. This project is currently in beta.

1.1.5 What is GNUstep's position towards KDE and the GNOME project?

You can use GNUstep with GNOME and/or KDE. GNUstep displays on top of X11. You can still do programming in C (since Objective-C is just a super-set of C), and when GCC gets around to it, you'll be able to mix C++ and Objective-C code in the same file.

GNUstep, is much more than a window manager or desktop environment. It frees you to develop cross-platform applications without the work of developing an OS independent framework from scratch. It gives you lots of basic functionality, from font panels to Unicode strings to distributed objects.

1.1.6 How can I get GNUstep?

Many distributions include packaged versions of GNUstep (Debian, etc). To compile from sratch, download the GNUstep Startup package or get the HOWTO from `gnustep-howto_toc.html`. Get the latest releases from `ftp://ftp.gnustep.org/pub/gnustep/core`.

1.1.7 How do you run GNUstep?

You are presumably under the misapprehension that GNUstep is some sort of program or window manager.

It isn't.

GNUstep is a whole load of things — primarily a set of libraries for developing software.

At present, it's those libraries, plus various command-line based support tools and service providing daemons, plus various GUI development tools, a GUI desktop/workspace application, etc.

At no stage will you ever 'run' GNUstep — you will run applications and tools and will make use of it's services. At some point you may well find packages distributed as 'GNUstep' systems in the way that you get 'GNU/Linux' systems packaged today. Look at Simply GNUstep `http://simplygnustep.sourceforge.net/` for instance.

If you want to see a sample GUI application running you need to build GNUstep and look at the example applications in the gnustep-examples package. Build 'Finger' or 'Ink' and start it with 'openapp Finger.app' or 'openapp Ink.app'

To look best, use WindowMaker (the currently preferred GNUstep window manager) as your window manager.

1.1.8 Is there a web site?

See `http://www.gnustep.org/`.

1.1.9 When is GNUstep intended to be available?

It's usable now. Major releases are made about every six months. However, if you are a serious developer, it's probably best to use the latest snapshots.

1.1.10 What is usable?

Most of GNUstep is quite usable and there are many complex applications that work well. However, GNUstep does not completely track the latest changes that Apple makes to their interface and there are still some parts that need some work).

What does this mean for users? Many applications will run quite well. Applications that require very complex text handling and some unusual features and/or some of the latest additions to Cocoa may not work as well.

1.2 Compiling and Installing

1.2.1 How do I compile GNUstep on my machine?

Read the file `GNUstep-HOWTO`, which comes with the GNUstep distribution (gnustep-make), and also is available separately on the GNUstep web site.

1.2.2 Are there any precompiled packages available?

Check `http://www.gnustep.org/resources/sources.html` for links to RPMs, Debian packages, and BSD ports. There's also Windows installers, Mac OS X binaries and others.

1.2.3 What are these type and size warnings?

These warnings:

```
/usr/bin/ld: warning: type and size of dynamic symbol
`__objc_class_name_NSConstantString' are not defined
```

are a common occurence and are due to a mismatch between gcc and ld. They don't do any harm so they can be safely ignored. They have been fixed in GCC version 3.1.

1.2.4 What are these import warnings?

Do you get this obnoxious warning whenever you compile an application, tool, or Objective-C program:

```
warning: using `#import' is not recommended
[...]
```

Up until gcc 3.4, the #import directive was not implemented correctly. As a result, the GCC compiler automatically emitted a warning whenever #import was used. As of gcc 3.4, this problem has been fixed, so presumably, this warning is no longer emitted when code is compiled. If you are using an early compiler, you can supress these warnings by adding `-Wno-import` to your include (cpp) flags.

1.3 Compatibility and Layout

1.3.1 Can I run NeXT OPENSTEP or Mac OS X programs on GNUstep?

You can't run these programs on GNUstep, but if you have the source code for the programs, you should be able to port them to GNUstep and compile them. Whether or not you will be able to run them depends on how complete GNUstep is at the time.

1.3.2 Is GNUstep following changes to OpenStep and Mac OS X?

Yes, gnustep-base already contains the documented changes in the Foundation library. GNUstep aims to be compatible with both the OpenStep specification and with Mac OS X. It should be easy to write an application that compiles cleanly under both GNUstep and Cocoa.

1.3.3 Do we have to have the NEXTSTEP look and feel?

GNUstep is aiming for something like the NEXTSTEP 3.3 look and feel. Although we don't want to force anyone into this, a lot of the power and ease of use comes from this feel. The look of GNUstep is something different — buttons and other widgets can look different but still act the same way. We hope to implement themes which will allow this.

1.3.4 What's up with the directory structure?

First of all, GNUstep uses a slightly different directory structure than NEXTSTEP or Mac OS X. Part of this is historical, part is because we can't do things the same way (see see

Section 1.3.5 [Why not use framework bundles?], page 4). Although currently the structure is very similar to the one used in Mac OS X.

1.3.5 Why not use framework bundles?

Framework bundles are much more difficult to port and to use, and are very unnatural on a UNIX system; extremely unnatural on Windows. In a framework bundle, the shared dynamic library is inside a framework wrapper directory. Because of this, the dynamic linker can't find it.

We have frameworks, so how do we work around that? Well, we build dynamic links from a directory inside the dynamic linker path into the framework, which work, but then you can't move the framework anywhere else on the system, otherwise you break the link, and nothing will find the framework any longer!

On systems without dynamic links, like Windows, we can't even do this! We have to copy the library from the framework into the dynamic linker path, but that is simply a shared library then! Absolutely *no* difference. You put the dynamic library in a system directory in the dynamic linker path, and associate with that library a resource directory.

OpenStep for Windows did that, and still called them frameworks. So we can do the same, and call our libraries frameworks.

In a shared library, the shared dynamic library is in a directory which is in the path to the dynamic linker. the dynamic linker can find it very easily. this is how all shared and static libraries work on UNIX systems, Windows systems and possibly most system at all.

Moreover, the OpenStep API requires us to provide some stuff for frameworks, like creating and registering automatically a framework object each time a framework is used (linked at runtime, or linked into the app), and attaching to it the list of classes inside the framework - which are not particularly trivial to implement — they depend on playing with the linker and the object file format — and might produce troubles when porting. And we never use these facilities.

For Apple Mac OS X sure it's easier. They can modify the system linker, compiler, the system dynamical linker. They always know on which platform they are working (their own), etc. They can modify the system to support frameworks natively. Easy that way.

But GNUstep is meant to run on many different platforms, platforms which we don't control (Windows, Sun Solaris, Darwin, GNU/Linux, UNIX system variants) and which have different linkers and do not support frameworks natively. On some systems it's difficult to just load a bundle or compile a shared library!

So building the core libraries as 'libraries' means that it's much easier to port them, and it's much more difficult to break them.

Sure, frameworks have a bundle of resources associated with it — but we can very easily associate a bundle of resource with a shared library, no reason why not. We are doing it.

So please note that GNUstep libraries are meant to be much similar to Mac OS X frameworks. They are composed of a shared library and associated with a bundle of resources. There is a difference in terminology, in where the resources are installed, and possibly a slight difference in the NSBundle API to get to the resource bundle (anyway, it's a one line difference between Mac OS X and GNUstep, so it looks like very easy to #ifdef).

In other words, GNUstep libraries are meant to basically do the same as frameworks do on Mac OS X, but to be portable to very different platforms (such as Windows).

1.4 Troubleshooting

1.4.1 Problems compiling (loading shared libs)

If you get something like

```
plmerge: error while loading shared libraries:
libgnustep-base.so.1: cannot open shared object file: No such file or directory
```

or this:

```
Making all for service example...
make[2]: *** [example.service/Resources/Info-gnustep.plist] Error 1
make[1]: *** [example.all.service.variables] Error 2
make[1]: Leaving directory '/home/multix/gnustep-gui-0.8.6/Tools'
make: *** [internal-all] Error 2
```

This means your GNU make is being overly protective. When you try to become root (e.g. to install something), certain environment variables like LD_LIBRARY_PATH are unset in order to reduce the possibility of a security breach. If you are becoming root to install something, you need to exec the GNUstep.sh file as root, just as you do when you login. Although for simplicity, you can also try this:

```
make LD_LIBRARY_PATH=$LD_LIBRARY_PATH
```

You could also be having problems with gcc. gcc 2.96 does not work (Mandrake 8.1, perhaps others). Use a different compiler, like gcc 3.x.

1.4.2 Problems compiling (GNUstep Internal Error)

If you get

```
GNUSTEP Internal Error:
The private GNUstep function to establish the argv and environment
variables was not called.
Please report the error to bug-gnustep@gnu.org.
```

when compiling the gui library, there could be several things wrong. One is that you installed the gnustep-objc library, but the compiler found another Objecive-C library (like the one that came with gcc). If you are using gcc 3.x, DO NOT use the gnustep-objc library.

There could also be a mismatch between the base and gui library versions. Make sure you have the latest release of each library installed.

1.4.3 Problems running tools and compiling

If you have a system that has SELinux enabled (Fedora Core for example), you may have trouble running and/or compiling (some tools are run during the compilation process) GNUstep. This is due to the use of ffcall and/or libffi and other techniques used to access memory in a way that SELinux does not like. You might get errors like

```
trampoline: cannot make memory executable
/bin/sh: line 5:  8427 Aborted                 ././shared_obj/
make_services --test GSspell.service/Resources/Info-gnustep.plist
gmake[2]: *** [GSspell.service/Resources/Info-gnustep.plist] Error 1
```

or

```
libgnustep-base.so.1.13: cannot restore segment prot after reloc: Permission denied
```

If you are using ffcall, you might need to switch to libffi. But in either case, it might help to do this:

```
chcon -t texrel_shlib_t /usr/GNUstep/System/Library/Libraries/*.so
```

after installing the base libraries.

1.4.4 Problems with gcc3

Don't forget you need to update binutils and libc also.

1.4.5 Problems with Alt key

It's possible the Alt key is not where you think it is or is defined incorrectly. Try running the GSTest application, KeyboardInput test (located in the examples package at `ftp://ftp.gnustep.org/pub/gnustep/core`) to test it. See `http://www.gnustep.org/resources/documentation/User/Gui/KeyboardSetup.html` for information on how to change the settings.

If you are using WindowMaker, it's possible it is grabing this key and using it for itself. To check, open Window Maker's WPrefs and go to the Mouse Preferences. Then use another value for the "Mouse grab modifier" (bottom right). That will allow you to alt-drag things.

1.4.6 Problems with fonts

Why do the characters get changed to asterisks ('*')?

The problem you are getting come from the fact that the xlib backend (when not using Xft) will only use one fixed X font for a given font name. If the font "helvetica" is used inside of GNUstep the one selected X font, in your case "-*-helvetica-medium-r-normal–12-*-*-*-p-*-iso8859-1" is used. So only characters (or glyphs) that are available in that font can be displayed. The selection of which font name to use happens inside the font_cacher and is more or less at random (the order fonts are listed by the X system).

You can influence the fonts that are available by setting:

```
defaults write NSGlobalDomain GSFontMask "*iso8859-13"
font_cacher
```

(or using a different character set, like iso8859-2). This is really a bug in GNUstep, but it hasn't been fixed yet.

The other option is the use the art backend, which handles fonts much better. When compiling gnustep-back, start with

```
./configure --enable-graphics=art
```

1.4.7 No characters displayed.

When using the xlib backend, no characters are displayed in any GNUstep applications.

The xlib backend has font anti-aliasing turned on by default. It's possible that GNUstep can't find any fonts on your system that can be properly anti- aliased. Try

```
defaults write NSGlobalDomain GSFontAntiAlias NO
```

to turn off font anti-aliasing.

1.4.8 No Makefile

I tried to compile something and I get:

```
GNUmakefile:27: /Makefiles/common.make: No such file or directory
GNUmakefile:39: /Makefiles/aggregate.make: No such file or directory
gmake: *** No rule to make target `/Makefiles/aggregate.make'.  Stop.
```

Make sure you have installed the gnustep-make package and also type:

```
source /usr/local/share/GNUstep/Makefiles/GNUstep.sh
```

You can put this line in your `.profile` or `.bash_profile` file so that it is done automatically when you log in.