# GNUstep Filesystem Hierarchy Document

Last Update: 19 November 2026

Authors: Nicola Pero, Tim Harrison, Martin Brecher, Adam Fedor, Richard Frith-Macdonald

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation.

# Table of Contents

1.1 The System Domain ..... 1
1.2 The Local Domain ..... 1
1.3 The Network Domain ..... 1
1.4 The Users Domain ..... 2
1.5 Structure of a Domain ..... 2
1.5.1 The GNUstep Filesystem Layout ..... 2
1.5.2 Accessing Domain Locations ..... 3
1.5.3 Applications ..... 4
1.5.4 Admin Applications ..... 4
1.5.5 Web Applications ..... 4
1.5.6 Tools ..... 5
1.5.7 Admin Tools ..... 5
1.5.8 Library ..... 6
1.5.9 Headers ..... 6
1.5.10 Libraries ..... 6
1.5.11 Documentation ..... 7
1.5.12 Documentation (Info) ..... 7
1.5.13 Documentation (Man Pages) ..... 7
1.5.14 Folders inside Library ..... 8
1.5.14.1 ApplicationSupport ..... 8
1.5.14.2 Bundles ..... 8
1.5.14.3 ColorPickers ..... 9
1.5.14.4 Colors ..... 9
1.5.14.5 DTDs ..... 9
1.5.14.6 DocTemplates ..... 9
1.5.14.7 Fonts ..... 9
1.5.14.8 Frameworks ..... 9
1.5.14.9 Images ..... 10
1.5.14.10 Libraries/Java ..... 10
1.5.14.11 Libraries/Resources ..... 10
1.5.14.12 KeyBindings ..... 10
1.5.14.13 PostScript ..... 10
1.5.14.14 Services ..... 11
1.5.14.15 Sounds ..... 11
1.5.14.16 Tools/Resources ..... 11
1.6 Configuration ..... 11
1.6.1 File Format ..... 12
1.6.2 Windows (MINGW) ..... 12

## 1.1 The System Domain

The System domain contains all files which were included in the default GNUstep installation or distribution. These files are normally managed by the distribution/packaging system used to install GNUstep; thus, making modifications to these files is highly discouraged. In addition, only the system administrator ('root' on most UNIX systems) should have permissions to write to that domain.

Normally you can expect to find gnustep-make and the basic GNUstep libraries (Foundation and AppKit) in this domain, and also essential system applications (the Workspace Manager, the Editor, applications related to system administrative tasks), developer applications (Project Center and Gorm, as well as header files), essential extensions (bundles for XML, SSL, RTF, etc), as well as all software installed by the manufacturer of your distribution.

In the GNUstep filesystem layout, the entire System domain is found in the `System` folder of the GNUstep installation.

## 1.2 The Local Domain

The Local domain is meant as the location for installing software which was not included with your GNUstep distribution and which you or your local sysadmin compile and/or install manually. These may include third party applications, custom extension libraries and their related header files, etc. Every software (except for gnustep-make, gnustep-base, gnustep-gui and gnustep-back which for historical reasons by default install into the System domain) should install by default into the Local domain, so that if you download a source tarball of the software and you install it, it installs by default in the right place for this operation (the Local domain). Distributions should override this default manually when they package the software they want to distribute as part of their distribution, so that in that case the software is installed in the System domain.

In the GNUstep filesystem layout the entire Local domain is installed as the `Local` folder of your GNUstep installation.

## 1.3 The Network Domain

The `Network` domain is optional and is usually coalesced with the `Local` domain by default; this is particularly appropriate for use on stand alone systems such as your home workstation. However, the Network domain can be of great use in networked, corporate environments. Its main purpose is to hold files exported from a central server in your network or from other workstations. Most times, remote directories containing applictations or general data used by several workstations in the network are mounted using the Network File System (NFS). Such usage gives administrators the possibility of providing application or resources to a vast number of workstations while only having to

manage the software in one place. This is especially useful when workstations are used by several users with different tasks and requirements. If you want to take advantage of the Network domain, you need to use a filesystem layout with a separate Network domain.

In the GNUstep filesystem layout the Network domain is the same as the Local domain; if you want to use the Network domain there is a separate GNUstep filesystem layout variant with a separate Network domain, in which case the entire Network domain is installed as the `Network` folder of your GNUstep installation.

## 1.4 The Users Domain

The main purpose of the Users domain is to hold GNUstep related files which shall not be available to other users on the system but only to the user owning them. This includes the GNUstep defaults database (which holds system settings, application preferences and customized resources) as well as temporary data related to services and file type associations for programs. It also holds user installed applications and tools (each user has the ability to install their own version of an application or tool).

In the GNUstep filesystem layout (and in most other layouts too) the User domain is completely contained in a subdirectory of the user's home directory called `GNUstep`.

## 1.5 Structure of a Domain

In this section we examine the interesting locations in a domain. We will enumerate the locations, and discuss what should be installed in each location, and how they are mapped to directories on disk in the GNUstep filesystem layout and in a general filesystem layout.

### 1.5.1 The GNUstep Filesystem Layout

We quickly present the GNUstep filesystem layout for a domain first because it is an essential reference for all discussions on the structure of a domain.

The GNUstep filesystem layout is the simplest layout, in which every domain is a directory on disk, and all locations in the domain are subdirectories of the domain.

In that case, a domain has the following structure on disk:

```
Domain/
       Applications/
       Applications/Admin/
       Defaults/   (User domain only)
       Library/
       Library/ApplicationSupport/
       Library/ApplicationSupport/Palettes
       Library/Bundles/
```

```
Library/Documentation/
Library/Documentation/info/
Library/Documentation/man/
Library/Frameworks/
Library/Headers/
Library/Libraries/
Library/Libraries/Java/
Library/Libraries/Resources/
Library/Makefiles/   (System domain only)
Library/Services/
Library/Tools/Resources/
Library/WebApplications/
Tools/
Tools/Admin/
```

The terminology for locations is derived from this filesystem layout, and it can be useful to use this directory structure as a reference point for all discussions. For example, every domain must have a 'Library' location.

### 1.5.2 Accessing Domain Locations

In order to install and run software that uses some resources, you need to be able to install the resources in the appropriate location, and your software needs to be able to locate these resources when it's running.

Since domain locations can be mapped to arbitrary locations on disk, you must use the appropriate gnustep-make and gnustep-base facilities to install things in the right place and to find things at runtime.

GNUstep-make creates makefile variables for all the domain locations. If you need to perform some custom installation for your software, you must use these variables to make sure your installation will work with all filesystem layouts. For example, the `Applications` location for the domain where the software will be installed is available as the `GNUSTEP_APPS` variable. You can also access the locations for specific domains by using the variables `GNUSTEP_SYSTEM_APPS`, `GNUSTEP_NETWORK_APPS`, `GNUSTEP_LOCAL_APPS` and `GNUSTEP_USER_APPS`.

GNUstep-base provides you with the `NSSearchPathForDirectoriesInDomains()` function that allows you to retrieve the domain locations at runtime. You must lookup resources only via this function. For example, the `Applications` location can be found by using the `NSApplicationDirectory` directory key, so you can use it in your software to iterate over all the `Applications` directories in the various domains searching for an application.

In general, all interesting domain locations have a set of variables defined in gnustep-make (such as `GNUSTEP_APPS`, `GNUSTEP_SYSTEM_APPS`, `GNUSTEP_NETWORK_APPS`, `GNUSTEP_LOCAL_APPS` and `GNUSTEP_USER_APPS`) and a corresponding directory key in gnustep-base (such as `NSApplicationDirectory`).

When examining the various domain locations, we will explicitly mention the gnustep-make variables and the gnustep-base directory keys that can be used to access them.

### 1.5.3 Applications

The `Applications` location contains applications. Applications are programs that typically have a GUI interface and contain associated resource files, such as images, localization files and other program elements.

Important applications which are part of GNUstep and which are often distributed as part of a core GNUstep distribution include:

```
Gorm.app
ProjectCenter.app
GWorkspace.app
Preferences.app
```

In GNUmakefiles, the `Applications` location is available via the `GNUSTEP_APPS` variable, which is the Applications location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_APPS`, `GNUSTEP_NETWORK_APPS`, `GNUSTEP_LOCAL_APPS` and `GNUSTEP_USER_APPS`.

In gnustep-base, the `Applications` locations are available by using the `NSApplicationDirectory` directory key for `NSSearchPathForDirectoriesInDomains()`.

### 1.5.4 Admin Applications

The `Admin Applications` location contains applications that are only useful to the system administrator. A normal user wouldn't have enough privileges to use these applications in a useful way.

In GNUmakefiles, the `Admin Applications` location is available via the `GNUSTEP_ADMIN_APPS` variable, which is the Admin Applications location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_ADMIN_APPS`, `GNUSTEP_NETWORK_ADMIN_APPS`, `GNUSTEP_LOCAL_ADMIN_APPS` and `GNUSTEP_USER_ADMIN_APPS`.

In gnustep-base, the `Admin Applications` locations are available by using the `NSAdminApplicationDirectory` directory key for `NSSearchPathForDirectoriesInDomains()`.

### 1.5.5 Web Applications

The `Web Applications` location contains web applications that were created using GSWeb or SOPE. These are programs contained with their resources in small wrappers very similar to standard applications.

In GNUmakefiles, the `Web Applications` location is available via the `GNUSTEP_WEB_APPS` variable, which is the Web Applications location for

the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables GNUSTEP_SYSTEM_WEB_APPS, GNUSTEP_NETWORK_WEB_APPS, GNUSTEP_LOCAL_WEB_APPS and GNUSTEP_USER_WEB_APPS.

In gnustep-base, the Web Applications locations are available by using the GSWebApplicationDirectory directory key for NSSearchPathForDirectoriesInDomains().

### 1.5.6 Tools

The Tools location contains tools and executable scripts. Tools are programs which generally have a command-line interface. Most are not meant to be used by the average user.

Important tools which are part of GNUstep and which are often distributed as part of a core GNUstep distribution include:

```
openapp
defaults
gdomap
gdnc
gpbs
```

In GNUmakefiles, the Tools location is available via the GNUSTEP_TOOLS variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables GNUSTEP_SYSTEM_TOOLS, GNUSTEP_NETWORK_TOOLS, GNUSTEP_LOCAL_TOOLS and GNUSTEP_USER_TOOLS.

In gnustep-base, the Tools locations are available by using the GSToolsDirectory directory key for NSSearchPathForDirectoriesInDomains().

### 1.5.7 Admin Tools

The Admin Tools location contains tools and executable scripts that are only useful to the system administrator. A normal user wouldn't have enough privileges to use these applications in a useful way.

In GNUmakefiles, the Admin Tools location is available via the GNUSTEP_ADMIN_TOOLS variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables GNUSTEP_SYSTEM_ADMIN_TOOLS, GNUSTEP_NETWORK_ADMIN_TOOLS, GNUSTEP_LOCAL_ADMIN_TOOLS and GNUSTEP_USER_ADMIN_TOOLS.

In gnustep-base, the Admin Tools locations are available by using the GSAdminToolsDirectory directory key for NSSearchPathForDirectoriesInDomains().

### 1.5.8 Library

The `Library` location contains most of the resources that are located and loaded at runtime. These resources are organized in folders (directories) inside `Library`; the most important `Library` folders will be described later on.

Like all systems inspired by OpenStep, resources are mostly organized in bundles and small wrappers that contain both machine-dependent files (such as executables or loadable object files) and general machine-independent resources (such as images or text files). For this reason, the `Library` location will contain both machine-dependent and machine-independent files.

The structure of the folders within Library is the same in all filesystem layouts, with a few exceptions: in the GNUstep filesystem layout, the `Libraries`, `Headers`, `Documentation` and `WebApplications` folders are all inside `Library`, but this is not necessarily true for other filesystem layouts.

Vice versa, it's common on other filesystem layouts (eg, FHS) to put `Applications` and `Admin Applications` as folders inside `Library`.

In GNUmakefiles, the `Library` location is available via the `GNUSTEP_LIBRARY` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_LIBRARY`, `GNUSTEP_NETWORK_LIBRARY`, `GNUSTEP_LOCAL_LIBRARY` and `GNUSTEP_USER_LIBRARY`.

In gnustep-base, the `Library` locations are available by using the `NSLibraryDirectory` directory key for `NSSearchPathForDirectoriesInDomains()`.

### 1.5.9 Headers

The `Headers` location contains header files associated with a library located in `Libraries`.

In GNUmakefiles, the `Headers` location is available via the `GNUSTEP_HEADERS` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_HEADERS`, `GNUSTEP_NETWORK_HEADERS`, `GNUSTEP_LOCAL_HEADERS` and `GNUSTEP_USER_HEADERS`.

In gnustep-base, the `Headers` location is not currently available.

### 1.5.10 Libraries

The `Libraries` location contains libraries (shared/static object files that are linked into programs). (NOTE: In the GNUstep filesystem layout, the Libraries directory being in Library may sound somewhat redundant, however, it could be read as "a Library of shared libraries").

In the GNUstep filesystem layout, the `Library/Libraries` folder contains two other folders: `Resources` and `Java`. It's important to notice that when the `Libraries` location is moved elsewhere, these folders

are not moved; they will still be in `Library/Libraries/Resources` and `Library/Libraries/Java`.

In GNUmakefiles, the `Libraries` location is available via the `GNUSTEP_LIBRARIES` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_LIBRARIES`, `GNUSTEP_NETWORK_LIBRARIES`, `GNUSTEP_LOCAL_LIBRARIES` and `GNUSTEP_USER_LIBRARIES`.

In gnustep-base, the `Libraries` locations are available by using the `GSLibrariesDirectory` directory key for `NSSearchPathForDirectoriesInDomains()`.

### 1.5.11 Documentation

The `Documentation` location contains documentation for libraries, applications, etc.

In GNUmakefiles, the `Documentation` location is available via the `GNUSTEP_DOC` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_DOC`, `GNUSTEP_NETWORK_DOC`, `GNUSTEP_LOCAL_DOC` and `GNUSTEP_USER_DOC`.

In gnustep-base, the `Documentation` locations are available by using the `NSDocumentationDirectory` directory key for `NSSearchPathForDirectoriesInDomains()`.

### 1.5.12 Documentation (Info)

The `Documentation (Info)` location contains documentation in info format.

In GNUmakefiles, the `Documentation (Info)` location is available via the `GNUSTEP_DOC_INFO` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_DOC_INFO`, `GNUSTEP_NETWORK_DOC_INFO`, `GNUSTEP_LOCAL_DOC_INFO` and `GNUSTEP_USER_DOC_INFO`.

In gnustep-base, the `Documentation (Info)` locations are not currently available.

### 1.5.13 Documentation (Man Pages)

The `Documentation (Man Pages)` location contains Unix man pages.

In GNUmakefiles, the `Documentation (Man Pages)` location is available via the `GNUSTEP_DOC_MAN` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_DOC_MAN`, `GNUSTEP_NETWORK_DOC_MAN`, `GNUSTEP_LOCAL_DOC_MAN` and `GNUSTEP_USER_DOC_MAN`.

In gnustep-base, the `Documentation (Man)` locations are not currently available.

### 1.5.14 Folders inside Library

In this section we discuss the standard folders that are available inside the `Library` location. To find any of these folders, just find the location of `Library` and then append the folder name (for example, in a GNUmakefile you can access the 'ColorPickers' folder of the installation domain as `$GNUSTEP_LIBRARY/ColorPickers`).

Some of the folders can also be accessed using direct variables, such as `GNUSTEP_BUNDLES`. You should prefer using these direct variables if you can because they are more future-proof (in case some of the folders become independent from `Library` in the future). All such cases should be documented here.

#### 1.5.14.1 ApplicationSupport

This directory contains bundles and other resources that are provided for an application, but that are not specifically distributed with that application. For instance, these may be third-party resources for an application.

For example, modules for the Preferences application may be stored here, in a `Preferences` subdirectory. In particular, Palettes for Gorm are stored in `ApplicationSupport/Palettes`.

In GNUmakefiles, this location is available via the `GNUSTEP_APPLICATION_SUPPORT` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_APPLICATION_SUPPORT`, `GNUSTEP_NETWORK_APPLICATION_SUPPORT`, `GNUSTEP_LOCAL_APPLICATION_SUPPORT` and `GNUSTEP_USER_APPLICATION_SUPPORT`.

In gnustep-base, the `ApplicationSupport` locations are available by using the `NSApplicationSupportDirectory` directory key for `NSSearchPathForDirectoriesInDomains()`.

#### 1.5.14.2 Bundles

This directory contains bundles. Bundles are collections of executable code and associated resources that may be loaded at runtime by an application or tool. Note: this directory is depreciated. Use ApplicationSupport to install bundles that can be used by an application.

In GNUmakefiles, this location is available via the `GNUSTEP_BUNDLES` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_BUNDLES`, `GNUSTEP_NETWORK_BUNDLES`, `GNUSTEP_LOCAL_BUNDLES` and `GNUSTEP_USER_BUNDLES`.

In gnustep-base, you can access the `Bundles` location as a folder inside the `Library` location.

#### 1.5.14.3 ColorPickers

This directory contains bundles that are used by the color picking system. They may include code that implements picking colors from a color wheel, a custom defined list of colors, etc.

This folder is accessed as the `ColorPickers` folder inside `Library`.

#### 1.5.14.4 Colors

This directory contains files that define specific color mappings for use within libraries or applications that require color definitions.

This folder is accessed as the `Colors` folder inside `Library`.

#### 1.5.14.5 DTDs

This directory contains any Document Type Definitions required for document parsing.

This folder is accessed as the `DTDs` folder inside `Library`.

#### 1.5.14.6 DocTemplates

This directory contains text templates for automatic documentation, as generated by autodoc. Any additional documentation template types must be placed in this directory, as a central location for documentation template types. Any templates installed within this directory must have an extension indicating what type of documentation system it is referenced by (ie. .gsdoc for the GNUstep implementation of autodoc).

This folder is accessed as the `DocTemplates` folder inside `Library`.

#### 1.5.14.7 Fonts

This directory contains fonts and files for organizing font information.

This folder is accessed as the `Fonts` folder inside `Library`.

#### 1.5.14.8 Frameworks

This directory contains frameworks. Frameworks are a type of bundle, which include, within their directory structure, a shared library providing a specific functionality (or group of related functionalities), and all resources required by that shared library.

All frameworks must have the extension `framework`, to indicate their usage.

Use of frameworks is generally discouraged, as it is difficult to support them in a clean way on multiple platforms. Bundles are a better method of organizing shared collections of resources and code.

In GNUmakefiles, this location is available via the `GNUSTEP_FRAMEWORKS` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly

by using the variables GNUSTEP_SYSTEM_FRAMEWORKS, GNUSTEP_NETWORK_FRAMEWORKS, GNUSTEP_LOCAL_FRAMEWORKS and GNUSTEP_USER_FRAMEWORKS.

In gnustep-base, the Frameworks locations are available by using the GSFrameworksDirectory directory key for NSSearchPathForDirectoriesInDomains().

### 1.5.14.9 Images

### 1.5.14.10 Libraries/Java

This directory contains Java classes. If you are using Java with GNUstep, you probably want to make sure these directories are in your CLASSPATH.

In GNUmakefiles, this location is available via the GNUSTEP_JAVA variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables GNUSTEP_SYSTEM_JAVA, GNUSTEP_NETWORK_JAVA, GNUSTEP_LOCAL_JAVA and GNUSTEP_USER_JAVA.

In gnustep-base, you can access the Libraries/Java location as the Libraries/Java folder inside the Library location.

### 1.5.14.11 Libraries/Resources

This directory contains resources used by shared libraries. In GNUstep a shared library can have an associated resource bundle (a bundle only composed of resources, with no object file), which is then installed into this directory.

For example, gnustep-base will get its resource bundle installed into GNUSTEP_SYSTEM_LIBRARY/Libraries/Resources/gnustep-base.

In GNUmakefiles, this location is available via the GNUSTEP_RESOURCES variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables GNUSTEP_SYSTEM_RESOURCES, GNUSTEP_NETWORK_RESOURCES, GNUSTEP_LOCAL_RESOURCES and GNUSTEP_USER_RESOURCES.

In gnustep-base, you can access the resource bundle associated with a library by using the [NSBundle +bundleForLibrary:] method (it is a GNUstep extension).

### 1.5.14.12 KeyBindings

### 1.5.14.13 PostScript

This directory contains directories for specific PostScript document types and definitions, allowing applications written using the GNUstep development environment to display PostScript documents, or communicate with printers using PostScript.

This folder is accessed as the PostScript folder inside Library.

#### 1.5.14.14 Services

This directory contains bundles that are specifically built to provide functionality between different programs (for example, spell checking, creation of a note from text within an email application). Services that are installed on the system must an extension of ".service".

In GNUmakefiles, this location is available via the `GNUSTEP_SERVICES` variable, which is the location for the domain in which the software will be installed. You can also reference the locations in the various domains directly by using the variables `GNUSTEP_SYSTEM_SERVICES`, `GNUSTEP_NETWORK_SERVICES`, `GNUSTEP_LOCAL_SERVICES` and `GNUSTEP_USER_SERVICES`.

In gnustep-base, you can access the `Services` location as a folder inside the `Library` location.

#### 1.5.14.15 Sounds

This directory contains sound files.

#### 1.5.14.16 Tools/Resources

This directory contains resources used by tools. In GNUstep a tool can have an associated resource bundle (a bundle only composed of resources, with no object file), which is then installed into this directory.

For example, a tool called `myTool` will get its resource bundle installed into `GNUSTEP_SYSTEM_LIBRARY/Tools/Resources/myTool`.

In GNUmakefiles, this location is available as the `Tools/Resources` folder inside the `Library` location.

In gnustep-base, you can access the resource bundle associated with your tool by using the `[NSBundle +mainBundle]` method (this semantic is a GNUstep extension).

### 1.6 Configuration

GNUstep supports arbitrary filesystem layouts to map the locations in the various domains to directories on the filesystem.

When you run gnustep-make's ./configure program you can use the –with-layout=xxx flag to select the filesystem layout that you prefer (choosing between the ones in the FilesystemLayouts directory, or creating your own in there!).

For most users, this is all they need to know.

In this section we'll go more into the details of how the filesystem layout system internally works; this is only useful if you need to do something advanced with it, typically because you have multiple GNUstep installations or you are trying to do some custom packaging of GNUstep.

The filesystem layout is determined by the GNUstep configuration file (or if that is not present, by default values built into the GNUstep make and base packages when they were configured and built).

The location of the GNUstep configuration file is built in to the make and base packages when they are configured using the –with-config-file option to the configure script. The path specified must be an absolute one for the make package, but may also be a path relative to the location of the base library itsself (as dynamically linked into applications) for the base package.

However, the location of the configuration file may also be specified using the GNUSTEP_CONFIG_FILE environment variable, overriding the value built in to the package, at any time when using the make package to build or install software. Support for the environment variable may also be enabled for the make package when its configure script is run.

The GNUSTEP_CONFIG_FILE environment variable is particularly useful if you have multiple installations and need to easily switch between them.

### 1.6.1 File Format

By default, the configuration file is called GNUstep.conf; it typically exists in /etc/GNUstep or /usr/local/etc/GNUstep (depending on how gnustep-make was configured) on a Unix-like system. This file is in a format suitable for being 'sourced' by the standard unix (Bourne) shell, consisting of lines of the form key=value, comments (everything on a line from the first hash (#) onwards), or blank lines.

This is very convenient on unix-like systems, but needs care for windows users. If a value contains whitespace or backslash characters (or the hash which would start a comment) it needs to be quoted by enclosing the whole value in single or double quotes. An alternative for values containing backslashes (the norm for a windows path) is to double up each backslash in an unquoted value.

The valid values for the keys in the GNUstep configuration file are documented in the GNUstep.conf file itself. Please check the GNUstep.conf.in file in your gnustep-make distribution for an up-to-date list of all the variables that you can change with explanations of what they do.

### 1.6.2 Windows (MINGW)

On ms-windows, for software development, you are likely to want to have an extra configuration file. This is because of the limitations of the make program (used to build and install software).

Basically the issue is that the make package doesn't really like the colons and backslashes in windows paths (using them is error prone) and can't tolerate whitespace in file names. So you want to do all the building in a unix-style environment using only unix-style paths.

On MSYS/MinGW this is done naturally by using the standard unix-style /usr/local/etc/GNUstep/GNUstep.conf config file, where the location is inside the MSYS unix-style emulation system. This is what is normally done by gnustep-make, so there is nothing special you need to do here.

On the other hand, the base library (and all applications since they are built using it) wants to work with native windows paths so that applications behave naturally as far as the end users are concerned, and therefore needs a configuration file containing windows-style paths rather than unix-like ones.

So, you need a different config file to be used by gnustep-base at runtime. And this is enabled by default – in fact gnustep-base will use ./GNUstep.conf as config file on MinGW, where the location is relative to the location of the gnustep-base.dll.

In other words, gnustep-make will use C:/xxx/usr/local/etc/GNUstep/GNUstep.conf (where 'xxx' is the MSYS installation path), while gnustep-base will use a GNUstep.conf file in the same directory as the gnustep-base.dll.

This ./GNUstep.conf file normally does not even exist; gnustep-base's ./configure will hardcode into gnustep-base.dll relative paths to all resources (relative from the installation location of gnustep-base.dll). If you modify the filesystem layout or relocate gnustep-base.dll, you should add a GNUstep.conf file with gnustep-base.dll that contains the relative locations of the directories (relative to the location of gnustep-base.dll).

It is recommended that this ./GNUstep.conf always contains relative paths to make relocation easier.