DESCRIPTION

This manual page is about cross-compilation and fat binaries. Fat binaries are packeges that you can supply which contains more the one binary of your e.g. application. So you can support multiple platforms with a single package.

To support this your system has to be built in a non-flattened way. Meaning that during the installation of *gnustep-make* you should have selected **−−disable-flattened** and the types of library combinations you want to support, through the **−−with-library-combo** option. With library combinations we mean the Objective-C runtime, the Foundation library and the Application library. For more details about this see the LIBRARY-COMBO section.

If you installed your **GNUstep** system in a non-flattened way all system dependend binaries are installed in subdirectories with *cpu/os/library-combo* information. That means for instance that the *gnustep-base* library will be installed in *Library/Libraries/ix86/linux/gnu−gnu−gnu/* when you are on an Intel x86 system, running linux with the GNU runtime for Objective-C and you installed **GNUstep**.

For each and every library-combo that you want to support you should create the environment through *gnustep-make*, because it installs a different *config.make* to support its own **CC**, **OPTFLAGS**, etc. flags.

LIBRARY-COMBO

An important issue is to let to a package the ability to deal with various libraries and configurations available now:

Objective-C runtimes
In the Objective-C world there are three major runtimes: the NeXT runtime, the Apple runtime and the GNU runtime (both with and without garbage collection enabled). They are different in several respects and a program or library that works at the runtime level should be aware of them.

Foundation libraries
There are several Foundation libraries an application or tool can be written on top of: NeXT Foundation library which runs on NeXTStep/OPENSTEP systems, gnustep-base, libFoundation and Apple Cocoa system.

Graphical interfaces
Until now three libraries provide or try to provide OpenStep compliant systems: the AppKit from NeXT, gnustep-gui and Cocoa from Apple.

If a program wants to work with all the possible combinations it will have to provide different binaries for each combination because it’s not possible to have a tool compiled for NeXT Foundation that runs with gnustep-base or vice-versa. To summarize, a program can be compiled for these combinations:

Objective-C runtime
nx (for NeXT), gnu (for GNU without garbage collection), gnugc (for GNU with garbage collection), apple (for Apple)

Foundation library
nx (for NeXT), gnu (for gnustep-base), fd (for libFoundation), apple (for Apple Cocoa)

GUI library
nx (for NeXT), gnu (for gnustep-gui), apple (for Apple Cocoa)

We’ll denote the fact that an application was compiled for a certain combination of the above values by using the abbreviations of the different subsystems and placing dashes between them. For example an application compiled for NeXT Foundation using NeXT AppKit will have the compile attribute nx−nx−nx. An application compiled for Apple Cocoa with the GNU compiler for Objective-C gnu−apple−apple and another one compiled for *gnustep-base* using *gnustep-gui* under Unix will be denoted by gnu−gnu−gnu. Here is a list of some of the possible combinations:

Runtime	Foundation	GUI
nx	nx	nx
nx	fd	gnu
gnu	gnu	gnu

gnu	fd	gnu
gnu	apple	apple
gnugc	gnu	gnu
gnugc	fd	gnu
gnugc	apple	apple
apple	apple	apple
apple	gnu	gnu

Note that one can choose his/her own packages to build; it is not required to have all the packages installed on the system. Not having all of them installed limits only the ability to build and distribute binaries for those missing combinations.

DIRECTORY STRUCTURE

For cross-compilation in a non-flattened directory structure is recommended, so that you can store on the same directory structure binaries for different machines. The standard **GNUstep** filesystem layout is normally used when a non-flattened directory structure is being used; this is obtained with the **−−with-layout=***gnustep* option when configuring *gnustep-make*. The entire **GNUstep** installation is then created inside */usr/GNUstep* (or another directory if you use the **−−prefix=...** option when configuring *gnustep-make*). Directories that contain binaries (such as the *Libraries* directory) inside */usr/GNUstep* are then set up to support fat binaries as follows:

```
Libraries/
  ix86/
    linux−gnu/
      gnu−gnu−gnu/
                    libgnustep−base.so
                    libgnustep−gui.so
      gnu−fd−gnu/
                    libFoundation.so
                    libgnustep−gui.so
```

To allow the right libraries to be found, you need to source *GNUstep.sh* before using **GNUstep**, and you need to start up your application by using **openapp**, which will locate the right binary for your library combo.

BUILDING FOR A LIBRARY-COMBO

The makefile package will allow the user to choose between different library combinations. To specify a combination you want to compile for just type:

```
  $ make library_combo=library-combo
```

For instance if you want to choose to compile using the GNUstep's Foundation implementation and use the GNUstep GUI library on a GNU/Linux machine you can do like this:

```
  $ make library_combo=gnu−gnu−gnu
```

If your project requires running configure before compiling there are two issues you have to keep in mind. 'configure' is used to determine the existence of particular header files and/or of some specific functionality in the system header files. This thing is usually done by creating a config.h file which contains a couple of defines like HAVE_... which say if the checked functionality is present or not.

Another usage of configure is to determine some specific libraries to link against to and/or some specific tools. A typical **GNUstep** program is not required to check for additional libraries because this step is done by the time the makefile package is installed. If the project still needs to check for additional libraries and/or tools, the recommended way is to output a *config.mak* file which is included by the main *GNUmakefile*, instead of using *Makefile.in* files which are modified by *configure*. The reason for not doing this is to avoid having the makefiles contain target dependencies like above, this way keeping only one makefile instead of several for each target machine.

The makefile package is written for GNU make because it provides some very powerful features that save time both in writing the package but also at runtime, when you compile a project.

BUILDING FOR AN ARCHITECTURE

In order to build a project for multiple architectures you'll need the development environment for the target machine installed on your machine. This includes a cross-compiler together with all the additional tools like the assembler and linker, the target header files and all the libraries you need.

The **GNUstep** makefile package should be able to compile and link an application for another machine just by typing

```
$ make target=target-triplet
```

where target-triplet is the canonical system name as reported by *config.guess*.

USING A LIBRARY-COMBO

When you use library-combos, you must always source *GNUstep.sh*. That allows you to switch library paths on the fly. If you want to switch to a different library-combo in your shell, and if you are using **bash**, it's common to first source *GNUstep-reset.sh* to reset all shell variables, then to source **GNUstep.sh** again. Let's assume we use gnu-gnu-gnu as our current **LIBRARY_COMBO** and we want to switch to gnugc−gnu−gnu, then we would use:

```
. /usr/GNUstep/System/Library/Makefiles/GNUstep-reset.sh
export LIBRARY_COMBO=gnugc−gnu−gnu
. /usr/GNUstep/System/Library/Makefiles/GNUstep.sh
```

SEE ALSO

debugapp(1), GNUstep(7), gnustep-config(1), openapp(1)

HISTORY

Work on gnustep-make started in 1997 by Scott Christley <scottc@net-community.com>.

Version 2.0.0 of gnustep-make introduced many changes with previous releases, which was mainly the work of Nicola Pero <nicola.pero@meta-innovation.com>

AUTHORS

This man-page was written by Dennis Leeuw <dleeuw@made-it.com> based on the DESIGN document from the gnustep-make source tree.

CREDITS

The DESIGN document was written by Ovidiu Predescu.

This work could only be as is due to the notes and corrects from Nicola Pero <nicola.pero@meta-innovation.com>.

COPYRIGHT