Using The MiscKit

Installation

Basics

It is recommended that you place the MiscKit source in /LocalDeveloper/Source. To install the MiscKit, simply type ªmake installº in a terminal window with the current directory at the MiscKit top level. This will build the MiscKit and install it. In order for the installation to proceed correctly, you should be root while installing the MiscKit. Other available Makefile targets are:

examples:	Builds the example apps; this is not done by the install process.
lib:	Builds just the MiscKit library.
bundles:	Builds the MiscKit bundle projects.
palettes:	Builds the MiscKit InterfaceBuilder palettes.
debug:	Builds and installs the MiscKit debug libraries.
all:	Builds the five targets listed above, without installing them.
uninstall:	Removes the MiscKit installation from your system.

distclean: Cleans all the subprojects (examples, bundles, libraries, and palettes) in the MiscKit.

Once you build and install the MiscKit, you can delete the source if you wish, since everything you need in order to use the MiscKit is installed into /LocalDeveloper.

Fat and thin libraries

The MiscKit by default builds fat, supporting the Intel and Motorola architectures. If you need to build a thin version of the MiscKit or otherwise change the target architectures, you will need to edit the top level Makefile. There are flags for each architecture (MOTOROLA_ARCH, INTEL_ARCH, HP_ARCH, etc.). All you need to do is comment out the architectures you don't want and uncomment the architectures that you do want.

The MiscKit has been successfully built on Motorola, Intel, and HP architectures using these directions.

Installation on 3.0 machines

If you are running NEXTSTEP 3.0: The MiscKit is developed to run on the most recent version of NEXTSTEP. It should compile without difficulty on either a 3.1 or a 3.2 machine. However, there are a few minor things you will have to do to get the MiscKit to build and install on a 3.0 machine. You need to do at least the following to compile for 3.0:

- In regexpr.c, change the declarations of malloc() and realloc() to have a return type of void * instead of char *.
- Set up to compile thin for m68k architecture. See the preceeding section for information about this.
- The top level Makefile attempts to index the installed documentation for Digital Librarian. In 3.0, this crashes, so you should remove the ixbuild call from the top level Makefile.
- In the Palettes, you will need to create the correct directories for the object files from the subprojects. For example, create the directory Palettes/MiscProgressView/obj/MiscProgressView.subproj before compiling the ProgressView palette. Any palette with a subproj in it needs this treatment.
- Also, the palettes which generate libraries such as the MiscProgressView need to have their Makefiles adjusted so that

they use the ªarº command instead of ªlibtool.º ªlibtoolº is only in NEXTSTEP 3.1 or higher, and is used so that fat libraries may be generated. Simply uncomment the 3.0 lines in Makefile.postamble and comment the 3.1/3.2 lines.

- The MiscCoolButtons palette seems to tickle a bug in 3.0 (either in NEXTSTEP or in Interface Builder). Apparently, when restarting IB after using this palette, you will get ªUncaught exception: NXReadOnlyString does not respond to -replaceWith:º. For the time being, deleting the file ~/.NeXT/defaults.nibd before restarting IB seems to alleviate the problem. We have no idea why this is happening; if you discover the problem let us know so that we can fix it. Under 3.x where x>0, this isn't a problem.

Using the MiscKit in a project

The MiscKit libraries are installed into the directory /LocalDeveloper/Libraries, which is not in the compiler's default search path for libraries. To add this directory to the compiler's search path, add this line to your project's Makefile.preamble:

OTHER_LDFLAGS = -L/LocalDeveloper/Libraries

Important Note: When you link against the MiscKit library, do not forget to use the -ObjC linker flag, which makes sure that all Objects and Categories are linked into your application. If you forget, you will get runtime errors which will crash your application whenever calling a method located in a category. If you are having trouble with the MiscString methods, this is the most likely reason for it. Forgetting the -ObjC linker flag is the most common cause of confused e-mail being sent to Don Yacktman. (On Don's off days, such e-mail will be replied to with the following message body: ªRTFMº.)

You can alternatively link against libmisckit.a or libmisckit_g.a, which are links placed in /usr/local/lib, if you prefer, and you won't need to add the OTHER_LDFLAGS above. However, if you use the palettes, the necessary libraries are only installed in /LocalDeveloper/Libraries, so you must either link them to counterparts in /usr/local/lib or add the OTHER_LDFLAGS as above. The headers to go with the palettes are in <misckit/x.h> but are not included by <misckit/misckit.h>. You should include any headers you need.

If you cannot get root permission on the machine you are using, you can still install and use the MiscKit, but you have to be a little bit more resourceful. First, edit the Makefile at the top level of the MiscKit and change the ROOT variable to the full path to your home directory. The MiscKit will install into your account in ~/LocalDeveloper and ~/usr/local/lib. Now, if you so desire, you can move the files around to where you prefer them to be, if it is different from where they are installed. (If you move the files, the uninstall target will not work, however.) In order to use the headers and libraries from their locations within your account, you will need to add lines like this to your project's Makefile.preamble so that the compiler can find them:

```
OTHER_CFLAGS = -I~/LocalDeveloper/Headers
OTHER_LDFLAGS = -L~/LocalDeveloper/Libraries
```