Using The MiscKit

Installing;¬Installation

Basics

It is recommended that you place the MiscKit source in /LocalDeveloper/Source. To install the MiscKit, simply type ªmake installº 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.
prep:	Tries to install tool.make and arch_tool if you don't have them.
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.
help:	Gives a summary of some important Makefile targets.

These are useful to MiscKit developers who need to recompile the libraries as they work on the kit:

build_core:	Builds the core libraries only. (ie, not palettes or bundles)
install_core:	Installs only the already built core libraries, if any.
_install:	Installs an already built kit without re-building everything.
build_debug_core:	build_core for the debug libraries. install_core will install whatever libraries have been built, be it debug, profile, and/or optimized.

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`. You may wish to keep copies of the top-level documentation (FAQ, Authors, etc.) for your own reference, however.

If you plan to keep the source, you can save about two Megabytes. After the install completes, you can delete the installed documentation and make it a link to the documentation in the MiscKit source distribution. You'll probably want to re-index it for Digital Librarian if you do that. You can also do the same for the Examples folder: delete it and make a link to the Examples folder in the MiscKit source distribution.

Fat;¬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. Remember that you can only compile for architectures that are supported by your installation of NEXTSTEP Developer. Specifically, if you thin your installation, you cannot built fat object files! Errors messages generated by an improper NS Developer installation can sometimes be misleading, so be forewarned.

Install3;¬Installation on 3.0 machines

The MiscKit is developed to run on the most recent version of NEXTSTEP. It should compile without difficulty on either a 3.1, 3.2, or 3.3 machine. However, there are many things you will have to do to get the MiscKit to build and install on a 3.0 machine. Since Don Yacktman does not have a 3.0 machine anymoreÐall his machines have been upgraded to 3.2 or 3.3Ðhe cannot test these fixes, and is therefore reluctant to put them in the Makefiles. Note that with the change to ProjectBuilder control you may not be able to build the kit anymore. The old Makefile is left in the Source directory as Makefile.old. You can replace the Makefile there with Makefile.old and cross your fingers that all will work OK or you can try to coerce the 3.0 Makefiles to work. The provided tool.make file needs to be installed as a bare minimum. According to Steve Hayman and Carl Lindberg you need to do at least the following to compile for 3.0:

(Note that the most recent kit releases haven't been built under 3.0, so you will likely need to do a lot more than this! If you discover more that needs doing, please let us know so that we can pass it on to others.)

- The top level Makefile attempts to do a sanity check using diff and otool; in 3.0 this apparently won't work due to differences in the archiving program. Just comment out the lines that attempt to do the diff.

- 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 regexpr.c, change the declarations of malloc() and realloc() to have a return type of void * instead of char *.

libtool1;¬· The MiscKit attempts to compile a fat library, which 3.0 cannot do. Because of this, you need to edit the Makefile in the top level so that all the XXX_ARCH flags are commented out. You should also edit the file Makefiles/lib/Makefile.programs and change the ARCHIVER program at the end of the file to be ªarº instead of ªlibtoolº.

libtool2;¬· 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.

tool.make;¬· The MiscKit moved to PB control in version 1.3.0. Since the Makefile for ªtoolº projects is only available for 3.2 and higher, you will need to upgrade or find your own ªtool.makeº to build the MiscKit. There is an attempt in the Makefiles directory of the MiscKit source distribution. The Makefile will attempt to install tool.make in the Makefiles directory if you don't have it, but will fail if you don't build as root or can't write to /NextDeveloper/Makefiles/app. There is no guarantee that this will work without the rest of the 3.2 Makefiles.

· 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.

arch_tool;¬· NEXTSTEP 3.0 doesn't have the /usr/lib/arch_tool program used by the Makefiles. You will find a bare-bones replacement in Examples/arch_tool that you can install in /usr/lib to get you by. It doesn't do everything arch_tool does, but it is good enough to get the MiscKit to compile. If you try to build the MiscKit, then the Makefile will attempt to install this program for you. It will fail if you are not root and you can't write to /usr/lib.

IBBug;¬· The palette projects need special subdirectories in the obj/ directories to be built before the compile so that they exist during the build. Any palette with a subproj in it needs this treatment. As an example, create the directory Palettes/MiscProgressView/obj/MiscProgressView.subproj before compiling the ProgressView palette. There is a shell script provided by Carl Lindberg to create the needed directories in Examples/mksubproj.

IBBug;¬· Having too many palettes available seems to tickle a bug in 3.0 (either in NEXTSTEP or in Interface Builder). The MiscKit FAQ;../FAQ.rtf;IBP;¬ has information about how to work around this problem.

Using the MiscKit in a project

Linking;¬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
```

ObjCFlag;¬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.

The headers to go with the palettes are in `<misckit/`*xxx*`.h>` but are not included by `<misckit/misckit.h>`. (The ªxxxº should be replaced with the name of the palettized object, of course.) You should include any

headers you need. There is a header, `<misckit/miscinterface.h>`, which attempts to include all the palette headers.

OtherRoot;¬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
```

Be sure to replace the ª~º with the actual path to your home directory since Makefile do not properly evaluate the ª~º, which is actually a shell feature and not a standard path specifier.

If you have troubles with some of the macros defined in the `<misckit/FREE.h>` or `<misckit/SELECT.h>` headers (they don't have the ªMISCº prefix so could possibly clash with your own favorite macros) you can define `MISC_SKIP_FREE` or `MISC_SKIP_SELECT` to skip those headers when importing `<misckit/MiscBase.h>` or `<misckit/misckit.h>`. Those flags will cause the inclusion of the offending header to be skipped. If you define these flags you won't be able to use the precompiled headers, though, unless you define the flags when precompiling them. The compiler will give you warnings, safe to ignore, if it is unable to use a precompiled header.

All trademarks used herin are owned by their respective owners. We're just borrowing them to give you a frame of reference by which you can understand what the MiscKit does.