Migration Toolbox Manual

Chapter 4

Kdarwin

A program to convert Khoros 1.0 vroutines into Khoros 2.0 kroutine objects

Chapter 4 - Kdarwin

A. kdarwin — *A program to evolve 1.0 programs to 2.0 objects*

Object Information

Category:	*Natural*
Subcategory:	*Evolution*
Operator:	*Migrate 1.0 vroutine*
Object Type:	*kroutine*
In Cantata:	*No*

Description

This program converts a 1.0 vroutine into a 2.0 kroutine object. It does 90% of the work required to bring an old Khoros 1.0 vroutine program into the new object-based software development system in Khoros 2.0. This program integrates all of the migration tools plus the Program Specification parsing code from 1.0 into a single program. The steps it follows while converting a program are:

1. Create a new kroutine software object in the specified toolbox under the specified object name. Uses the object name for the icon name if [-icon] was not specified.

2. Parse the *.prog file for the vroutine's code and documentation sections.

3. Call uisupdate to convert the *.pane file specified to be Khoros 2.0 compatible.

4. Call kfixpane to modify the Khoros 2.0 UIS file output by uisupdate so that the pane will be compliant with the Khoros 2.0 GUI conventions. This includes updating the short description of the pane (uses the program short description if [-idesc] was not specified), modifying the -R line to hold the fullpath to the binary, and changing the -H line to point to the new help directory.

5. Update the code and documentation files of the new kroutine with the sections from Khoros 1.0 *.prog file. This includes updating the use of the CLUI Information structure to the new name, *clui_info* which is the name used under Khoros 2.0. It also inserts code in the appropriate sections to set a variable called *program* in the main which used to be automatically defined in Khoros 1.0.

6. Call kfix on the l*oname*.c, *oname*.c, and *oname*.h to convert the Khoros 1.0 and non POSIX library calls to Khoros 2.0 library calls.

7. Call mkproto to get the ANSI prototype of the l*oname* routine, and insert it into *loname*.h public include file.

8. Attempt to run the gnu utility protoize to convert the l*oname*.c file from K & R C code into ANSI C code. Note, this usually doesn't work the first time, and may not ever work if you do not use the GNU gcc compiler for your compiles. However, usually changing your code from K & R to ANSI is just a matter a changing the declaration.

Though this program does most of the work necessary to convert a 1.0 vroutine into a working kroutine under Khoros 2.0, several steps must be taken by the user to finish the conversion. The code, the documentation, and the uis file of the new object will have to be checked for problems in formatting and checked for sections the migration tools could not handle. The following is a list of things that will have to be dealt with by hand after running kdarwin.

1. The man 3 sections of the Khoros 1.0 program specification will be put into their appropriate section of the routine header in the l*oname*.c file. However, all the roff formatting macros will still be there. So these macros must be removed, and the text should be reformatted to look like a proper Khoros 2.0 public routine header.

2. All files that were kfix'ed need to be checked for any sets of four # characters in a row. This sign followed by some text indicates that kfix knows the line needs a change, but is not sure how to accomplish the change. Since kfix uses sed to match sections of code, the more complicated C syntax means that it cannot always figure out how to make a change. For example, C function calls that span multiple lines can some times result in this type of error. Also, if a call to a function is used as a parameter in another function call and one or both of those function calls must change, kfix usually cannot figure which parameters belong with each function call. In such a case, you will need to change the code to the correct call, and rearrange the parameters according to the man page. Use kman to find the appropriate format for the call to the Khoros 2.0 function.

3. In Khoros 1.0 many programs declared external routines that they called in side the variable declaration section of a routine. This practice is bad under ANSI C because the extra declaration is made without the correct prototype causing the correct prototype to be overridden. So, all of these declarations should be removed.

4. Khoros 2.0 depends on ANSI compliance, so any routines converted must use the ANSI declaration format instead of the K & R one. kdarwin calls the GNU program protoize, if it exists, to automate this process. However, this process usually fails due to syntax errors in the code it is trying to convert. If protoize does fail, it will inform you, and give you the proper command to run protoize on the library source code file again. You will first have to correct the syntax errors in the file before you can run protoize again.

5. Finally, the Khoros 1.0 calls to manipulate the xvimage (1.0 VIFF) structure have been ported to 2.0 and put in the RETRO toolbox in the vutils and verror library objects. So, in order to get the library calls *creatimage*, *readimage*, and *writeimage*, the toolbox that holds your converted programs will need to include the RETRO public include file, and link against the libraries vipl, verror, and vutils in RETRO. See the Toolbox Programming Manual for instructions on how to makeone toolbox dependent on another.

Note that *readimage* has been rewritten in 2.0 to read all data types supported by data services, and forces the data and attributes into an xvimage structure. This fact implies that these programs can manipulate data as in 1.0. However, it also means that you lose the benefits that data services provides, like large data set handling, polymorphic data model, and data propagation. Also, you can lose data if the original format has extra data that does not map nicely into the Khoros 1.0 VIFF structure. The Khoros 2.0 VIFF is such a case, because it is capable of hold 5D data and it can contain extra data segments. The final limitation is that *writeimage* always outputs the Khoros 1.0 VIFF or xvimage data format.

We encourage you to use kdarwin to quickly move your contributed routines into the Khoros 2.0 environment; however, it would be best if you follow this quick conversion with a proper re-implementation of the operators using Data Services and one of the available Application Services.

Examples

Assume that you want to convert the Khoros 1.0 vroutine vspatial to Khoros 2.0, and place it in the Khoros 2.0 toolbox MYTB. Also assume you want to convert the l*.c file to a library object called my_ipl and that your Khoros 1.0 source tree is located in /usr/khoros. The following command tells kdarwin to do this. Note that the library my_ipl must exist **BEFORE** you run kdarwin. Use craftsman to create this library object.

```
% kdarwin -tb mytb -oname vspatial -lname my_ipl -cat "Image Proc"
-subcat "Classification (K1)" -icon "Spatial Analysis"
-description "Compute Spatial Feature Operators (NxM Window)"
-prog /usr/khoros/src/vipl/feature/vspatial/vspatial.prog
-pane /usr/khoros/repos/cantata/subforms/feature/vspatial.pane
```

The output from kdarwin will be as follows:

```
kdarwin: updating object database [MYTB::my_ipl]
kdarwin: Generating Imakefile & Makefile in src
kdarwin: Generating directory Imakefile & Makefile
kdarwin: updating object database [MYTB::my_ipl]
kdarwin: Generating Imakefile & Makefile in src
kdarwin: Generating directory Imakefile & Makefile
Generating empty ghostwriter files
kdarwin: done generating vspatial man page
kdarwin: done generating vspatial.h
kdarwin: done generating vspatial.c
kdarwin: done generating usage.c
kdarwin: done generating lvspatial.c
kdarwin: done generating vspatial help page
Get PROG information
Update UIS file to 2.0 format
Update UIS to comply with cantata standard format
Regenerate command line UIS code
kdarwin: done generating vspatial man page
kdarwin: done generating vspatial.h
kdarwin: done generating vspatial.c
kdarwin: done generating usage.c
kdarwin: done generating lvspatial.c
kdarwin: done generating vspatial help page
Update sections of object from prog spec
```

```
-- Convert 'vspatial->' to 'clui_info->' in prog sections
-- Update the program related prog sections
-- Update the library related prog sections
Run kfix lvspatial.c
```

Toolbox: MIGRATION
Program: kdarwin
Protoize of 'lvspatial.c' failed. This implies that the code still will not compile because of some cases that the automatic update routines cannot handle. You must fix these problems, and re-run protoize by hand. This can be accomplished by cd'ing to the src directory for the library that contains ´lvspatial.c´, and typing:

protoize -c "-fstrength-reduce -fpcc-struct-return -fwritable-strings -ansi -I/research/devel/mirage/work/mytb/include -I/research/devel/mirage/khoros/include -I/research/devel/mirage/retro/include -I/usr/local/motif/include -D__sun -D_POSIX_SOURCE" $MYTB/objects/library/my_ipl/src/lvspatial.c
Keep fixing problems and reprotoizing until it produces a lvspatial.c.sav file.
Update the man page database for the library
kdarwin: Finished Generating a man page for the function ´lvspatial´ in ´$MYTB/objects/library/my_ipl/man/lvspatial´

```
kdarwin: done generating my_ipl library headers
kdarwin: done generating my_ipl man page
Run kfix vspatial.c
Run kfix vspatial.h
kdarwin: done generating vspatial man page
kdarwin: done generating vspatial.h
kdarwin: done generating vspatial.c
kdarwin: done generating usage.c
kdarwin: done generating lvspatial.c
kdarwin: done generating vspatial help page
kdarwin: updating object database [MYTB::vspatial]
kdarwin: Generating Imakefile & Makefile in src
```

This initially converts the routine. After you have fixed the syntax problems and prototypes as indicated above in the above example, you can compile this program. Note that in order to compile, you need to make you're toolbox dependent on the RETRO toolbox. This only needs to be done once, and the resulting mytb.def file should look similar to the following one. The path to the retro toolbox will differ as per your specific site's installation. Note that the order on the TOOLBOX_LIBRARIES line for the retro libraries is important. Also, libraries that exist within mytb should be listed **BEFORE** the retro libraries.

```
% cat `kecho -echo path -tb mytb`/repos/config/imake_conf/mytb.def
KCOM mytb.def
KCOM toolbox configuration file for toolbox mytb

/*
```

```
 *  This file is included in the Makefile after Khoros.tmpl,
 *  Site.def, Library.def file and just before the Imakefile.
 *  This file can be modified to define other Toolbox definitions.
 */
#include <design.def>


/* assign the TB_NAME variable to the name of your particular toolbox.
 * By default the name of your toolbox is "MYTB".  Note, the toolbox
 * name MUST be in UPPER CASE.
 */
TB_NAME = MYTB
RETRO = /research/devel/mirage/retro


/* Set locations of bin and lib directories in your particular toolbox.
 * The default is $MYTB/bin and $MYTB/lib.
 */
LIBDIR = $(TB_NAME)/lib
BINDIR = $(TB_NAME)/bin


/* Set location of include files for your toolbox. You should not have
 * to change this symbol.
 * The default is $MYTB/include.
 * Note, /usr/include and $MYTB/include will automatically be
 * searched. You do not have to specify these.
 */
TOOLBOXINCLUDE = -I$(TB_NAME)/include -I$(RETRO)/include


/* Set library loading search path. You should not have
 * to change this symbol.
 * The default is $MYTB/LIBDIR.
 * Note, /usr/lib and $MYTB/lib will automatically be searched.
 * You do not have to specify these.
 */
TOOLBOXLIBDIR = -L$(LIBDIR) -L$(RETRO)/lib


/* Any special defines for your toolbox (eg, #ifdef DEBUG)
 * can be set here (eg, -DDEBUG)
 */
TOOLBOXDEFINES = /**/


/* List C libraries created (by you) in the toolbox.
 * These are vroutine or utility libraries. Note, each library
 * in the list must be preceded by a -l.
 * (eg. TOOLBOX_LIBRARIES = -lutil -lfrogs
```

```
 *
 * Also, list the full path to the libraries for the
 * TOOLBOX_DEP_LIBRARIES symbol. This allows your programs to depend
 * on a library. This is if a library changes, then make will know
 * to reload your program. An example of how to set this symbol is:
 * TOOLBOX_DEP_LIBRARIES = $(LIBDIR)/libutil.a $(LIBDIR)/libfrogs.a
 */

TOOLBOX_LIBRARIES = -lvipl -lverr -lvu
TOOLBOX_DEP_LIBRARIES = /**/


/* List all FORTRAN libraries created (by you) in the toolbox.
 * follow the same concept for listing the libraries as described
 * for the TOOLBOX_LIBRARIES and TOOLBOX_DEP_LIBRARIES above.
 */

FOR_TOOLBOX_LIBRARIES = /**/
FOR_TOOLBOX_DEP_LIBRARIES = /**/


/* List Xlib or X Toolkit dependent libaries created (by you) in the toolbox.
 * follow the same concept for listing the libraries as described
 * for the TOOLBOX_LIBRARIES and TOOLBOX_DEP_LIBRARIES above.
 */
X_TOOLBOX_LIBRARIES = /**/
X_TOOLBOX_DEP_LIBRARIES = /**/

/*
 * You may override any symbols previous defined in the imake system if
 * you wish. You should never have to override any sybmols.
 */

%
```

See Also

ghostwriter(1), kfix(1), kfixpane(1), mkproto(1), uisupdate(1), protoize(1)

Restrictions

This program was specifically designed to convert from vroutines to kroutines, so using it on any other program type will produce strange results.

References

Migration Toolbox Manual, Toolbox Programming Manual

Command Line Arguments

```
Usage for kdarwin: A program to evolve 1.0 programs to 2.0 objects
```

```
% kdarwin
  -tb             (string) Toolbox object to hold the new software object
  -oname          (string) Software object name to use the for the converted vroutine
  -cat            (string) Category Name used in Cantata
  -subcat         (string) Subcategory Name used in Cantata
  -description    (string) The short description of the object
  -icon           (string) Icon Name used for glyph in Cantata

  Required Mutually Exclusive Group - Specify exactly one of the 2 options:

    [Optional Mutually Inclusive Group - Specify all or none of the 2 options:]
      [-prog] (infile) Khoros 1.0 Prog filename to get program code and documentation information from
              (default = null)
      [-pane] (infile) Khoros 1.0 Pane Filename to get program uis information from
              (default = null)
    [-conf] (infile) Khoros 1.0 conf filename to use to get paths for *.pane and *.prog
            (default = null)

  [-lname]        (string) Library object to put the vroutine l*.c file into
                  (default = null)
```

This page left intentionally blank

Table of Contents

This page left intentionally blank