Think Class Library 2.0
CodeWarrior Port Package

Version 1.3 • 30 May 1994
Produced by Eric Scouten

Note
This document was produced using Microsoft Word and the TrueType fonts Palatino, Helvetica, Courier, and Zapf Dingbats. There are two versions of this document: `**README.word` which is a Microsoft Word 5 document, and `**README.standalone` which is a combined application/document created with Print2Pict. ❖

Contents

About the TCL Port Package

Introduction

This document describes the TCL 2.0 CodeWarrior Port Package, a set of files which will enable you to compile the Think Class Library version 2.0 with the Metrowerks CodeWarrior 68K and PowerPC compilers. Read this document carefully **before** using the package.

This package will be updated regularly as new versions of the Metrowerks compilers become available and as new bug fixes or features are incorporated. The most recent version should always be available for FTP on the site `daemon.ncsa.uiuc.edu` in the directory `/pub/TCL/contributors/Eric_Scouten`. Notification of these updates will be posted to the Internet newsgroup `comp.sys.mac.oop.tcl`.

This package **does not include** the Think Class Library itself, in either original or modified form. You must purchase a copy of Symantec C++ for Macintosh, version 7.0, in order to use the TCL.

Note
Additional information may be contained in a file named *`README.extra` in the root folder of the package. ❖

Package Requirements

To make effective use of the CodeWarrior Port Package, you need the following items:

- Symantec C++ for Macintosh, version 7.0.
- Metrowerks CodeWarrior Bronze, Silver, or Gold, DR/2 or newer.
- At least 30MB disk space available on your hard drive.
- A 680x0 Macintosh with at least 8MB of RAM, or a PowerPC Macintosh with at least 16MB of RAM.
- An unmodified copy of the Think Class Library, version 2.0.2. You will need the free updater provided by Symantec to upgrade the TCL from version 2.0 to version 2.0.2. This is available for anonymous FTP from the site `devtools.symantec.com` as part of the Symantec C++ updater to version 7.0.2. (It may also be available on other FTP sites.)

The following items **will not** be useful:

- Think C or Symantec C++, version 6.0 or earlier.

- Think Class Library, version 1.1.3 or earlier. (If you are using TCL 1.1.3, you may wish to investigate the PTR-TCL package provided by Jon Wätte on common FTP sites.)
- The free updater provided by Symantec to upgrade Symantec C++ from version 6.0 to version 7.0. The updater does not include the TCL version 2.0.
- Symantec's PowerPC Cross Development Kit. The port package will not work properly with the TCL version 2.0.1 included with the CDK.

Overview

Using the TCL CodeWarrior Port Package entails the following steps:

- Converting the Think Class Library to a form usable by the Metrowerks compilers.
- Converting your source code to a form usable by the Metrowerks compilers.

The following chapters of this document describe these steps in detail. You should also review the chapter entitled, "Using the Modified TCL," which describes the differences between the standard TCL used for the Symantec compiler and the modified TCL used for the Metrowerks compilers.

For More Information

If you have questions about using the TCL CodeWarrior Port Package, or about the TCL 2.0 in general, you may contact me at the following addresses:

Internet	scouten@maroon.tc.umn.edu
Telephone	+1 612 626 0746
Fax	+1 612 626 8131
US mail	Eric Scouten PO Box 13536 Minneapolis, MN 55414

Note

These addresses are valid through the end of **July 1994** only. After that time, I will be moving to Urbana, Illinois (University of Illinois). I do not yet have new addresses to provide. ❖

I am starting an unofficial list of bugs found in the TCL 2.0 which will be posted regularly on the Internet newsgroup `comp.sys.mac.oop.tcl`. If you discover a bug in the TCL (whether or not it is related to the CodeWarrior port), please send me an e-mail message describing the nature of the problem so that I may include it in the bug list (and correct it in the CodeWarrior port).

Please note that this package is provided **free of charge.** Although I am generally available (and interested) to talk about this package and any related topic, I may from time to time be unable to respond to requests for support of the package due to other commitments.

Creating the Modified TCL Source Code

In this stage, you will create a second copy of the Think Class Library source code on your hard drive. You should keep two copies of the TCL source code on your drive: the original source for use when compiling with Symantec's compiler, and the modified source for use with CodeWarrior. (The TCL source and precompiled headers take approximately 6MB.)

❖ **W A R N I N G**
It is absolutely important that you begin with the **unmodified** source code from the TCL 2.0.2. If you have modified the TCL yourself, make a copy of your changes and reinstall the TCL from your Symantec C+ 7.0 master disks. Then update the TCL source with the updater files provided by Symantec to version 2.0.2. (If later versions of the TCL are released, **do not** use these versions either.) ❖

Converting the updated TCL requires the following steps:

- Make a copy of the Think Class Library 2.0 folder. Place this folder **outside** the CodeWarrior compiler folder (and outside the Symantec C++ folder as well.)
- Open the MultiDiff application. Select "Apply Diff..." from the File menu. A standard open box will appear. Select the file `*TCL source.diff`. You will be prompted for the top folder to apply the patches to. Select the folder which contains the copy of the TCL you just made. Note that the changes are made **in place;** thus it is important to make a copy of the TCL prior to applying the patches.
- Wait. There is no way to stop the patcher once it's running. It will take several minutes to make the changes. (On a Centris 610 it takes about 7 minutes.)
- Copy the Templates folder from the your modified TCL folder to the TO MUNGE folder inside the Template Munger folder of the port package. Open the Template Munger application. It will run for a few moments and quit.
- Open the TEMPLATE HEADERS folder. Copy the files in this folder back to the folder where you are storing the modified TCL. (Any location inside the TCL folder will do.)
- Copy the header files `THINK.h` from your Symantec C++ folder to your CodeWarrior folder. If you maintain separate folders for 68K and PPC compilers, copy the header to both folders. (`THINK.h` can be found in the path `:Symantec C++ for Macintosh:Mac #includes:THINK #includes`.)

- Copy the header files `SANE.h` from your Symantec C++ folder to your CodeWarrior folder. If you maintain separate folders for 68K and PPC compilers, copy the header to the 68K compiler folder only. **Rename** the header to `ThinkSANE.h`. (`SANE.h` can be found in the path `:Symantec C++ for Macintosh:Mac #includes:THINK #includes`.)
- You may want to use the MultiDiff application to patch one or more the demonstration programs provided with the TCL. Project files for some of the demo programs are included (with binaries removed).

Creating the Precompiled Headers

You will need to precompile new header files for use with the CodeWarrior compilers. Doing so involves the following steps:

- Open the CodeWarrior project file `CW TCLHeaders68K.µ`.
- If you are using CodeWarrior DR/2, look for the line that reads:

  ```
  #define TCL_CW_VERSION 3
  ```

  Change the 3 to a 2. If you are using a newer version of CodeWarrior, see the comments in the source code to see if newer versions are supported.
- Open the Preferences... dialog, Access Paths panel. Add an access path to the folder where you placed the modified TCL source.
- Precompile the header file. It should take about two minutes. When it is complete, save the file as `CW TCLHeaders68K` (preferably in the same folder as your TCL source).

Repeat the same steps for the PowerPC compiler, substituting `PPC` for `68K` in the filenames listed above. The precompiled header files should take slightly more than 600K each on your hard drive.

Creating a Project File

You will find it easiest to start from the project files for the Art Class demo which was included with the port package. These projects loosely follow the segmentation in the sample TCL project files provided by Symantec. The first segment contains source files specific to the demonstration application. Remove these files and insert your own source files.

Building a New Project File

If you choose to create your own project file, you will need to be aware of a few considerations. The virtual function tables generated by the TCL tend to be rather large. For small TCL projects, you will be able to compile “as is.” If you have more than a few of your own classes, you will need to turn on “Far virtual function tables” in the Preferences Language panel.

To create the list of source files, you may simply include all of the files in the modified TCL source directory. Please note the following differences between the CodeWarrior projects and Symantec C++ projects:

Removed Source Files

```
CSaver.cpp
CStack.cp
PutObject1.cpp
BRLib.π
```

Added Source Files

```
BRClaInf.cpp
BRPriStr.cpp
UDebugging.cp    (from Metrowerks PowerPlant)
UExceptions.cp   (from Metrowerks PowerPlant)
```

Resource File Management

CodeWarrior's development environment does not include resource file management like that in Symantec C++. I have found it simplest to rely on Symantec C++ to perform resource management. CodeWarrior will resolve aliases when looking for the `<projectname>.rsrc` file; therefore you can alias your Symantec C++ resource file and give it the name of the CodeWarrior resource file.

Changes in Coding Technique

This section describes some of the differences between the Symantec C++ version of the TCL and the modified TCL for CodeWarrior. The compatibility issues you will face fall primarily into two categories: C++ templates and exception handling.

C++ Templates

As of the current release (DR/3), CodeWarrior does not support C++ templates. As it turns out, the TCL 2.0 makes fairly light use of templates – only the classes `CPtrArray`, `CList`, `CPtrArrayIterator`, and `CSaver` are templated classes. In addition, `CStream` has some templated helper functions. At present, the port package provides macros to replace the templates for all of these classes.

Two macros are defined for each templated class: one to declare the class (placed in a header file) and a second to define it (placed in a source file). These macros are triggered by a macro defined in the `CW HeadersTCL.h` file: (When CodeWarrior supports templates, you may comment out this line.)

```
#define TCL_NO_TEMPLATES
```

In general, the template workaround works as follows: A class defined as `template<class T> class CTemplate<T>` is instead declared as `class CTemplate_T`.

The macros to declare the templated classes (used in header files) are:

CList<T>	`TM_DECLARE_CList(T)`
CListIterator<T>	`TM_DECLARE_CListIterator(T)`
CPtrArray<T>	`TM_DECLARE_CPtrArray(T)`
CPtrArrayIterator<T>	`TM_DECLARE_CPtrArrayIterator(T)`
CSaver<T>	`TM_DECLARE_CSaver(T)`
CStream	`CW_Decl_StreamCalls(T)`

The macros to define the templated class (used in source files) are:

CList<T>	`TM_DEFINE_CList(T)`
CListIterator<T>	`TM_DEFINE_CListIterator(T)`
CPtrArray<T>	`TM_DEFINE_CPtrArray(T)`
CPtrArrayIterator<T>	`TM_DEFINE_CPtrArrayIterator(T)`
CSaver<T>	`TM_DEFINE_CSaver(T)`
CStream	`CW_Inst_StreamCalls(T)`

As a practical example of how to use the substituted template class, examine the list of collaborators maintained by `CCollaborator`. In `CCollaborator.h` the following lines have been added (flagged by `•• CW TCL` comments):

```
#ifdef TCL_NO_TEMPLATES                 // •• CW TCL
#define CCollaboratorList CPtrArray_CCollaborator
#endif                                  // •• CW TCL

class CCollaboratorList;
```

The original source code merely forward referenced a class named `CCollaboratorList`. This class was defined in `CCollaborator.cp` as a subclass of `CPtrArray<CCollaborator>`. In the code above , `CCollaboratorList` is `#define`d to have the same name as the macro template class – this class is then forward referenced by the same line of code as the original code used.

Here's the modified declaration from `CCollaborator.cp` :

```
#ifdef TCL_NO_TEMPLATES                 // •• CW TCL
TM_DECLARE_CPtrArray(CCollaborator)     // •• CW TCL
TM_DEFINE_CPtrArray(CCollaborator)      // •• CW TCL
#else                                   // •• CW TCL
struct CCollaboratorList : CPtrArray<CCollaborator>
{
};
#endif                                  // •• CW TCL
```

The macros here both define and instantiate the stand-in template class `CPtrArray_CCollaborator`.

Using Your Own Templates

You may use the Template Munger application to develop macro substitutions for your own templated classes. Doing so involves the following steps:

- Add the following lines to the header file which declares your templated class:

```
#ifdef TCL_NO_TEMPLATES
#include "<cl>.tm.h"  // __TEMPLATE__ <cl>.tm.h <cl>.tem
#else
    // standard template definitions
#endif
```

The __TEMPLATE__ comment is a special comment which triggers the Template Munger's macro writing behavior. It must be contained in a line-comment (//-style comment) and must be followed by one or two filenames. The first filename (required) is the name of the file to which the macro definitions will be written. (It should be the same as the file which is #included.) The second filename (optional) is the name of the source file which defines the methodsfor the templated class. (In the TCL, this file is typically given the suffix .tem.)

- Copy all of the header files and source files for your templated classes to the TO MUNGE folder inside the Template Munger folder of the port package.
- Make sure there is a TEMPLATE HEADERS folder in the Template Munger folder. It should have no files in it. If there are files, move them out of the way or delete them.
- Open the Template Munger application. It will run for a few moments and quit.
- Open the TEMPLATE HEADERS folder. Copy the files in this folder back to your project folder. These files will have the names you designated in the __TEMPLATE__ macro.

For each templated class it encounters, the Template Munger writes two macros to the designated file. The first is TM_DECLARE_<classname>(T). This macro contains the class definition which replaces template<class T> classname<t>. The second is TM_DEFINE_<classname>(T). This macro only appears if you declared a source file on the __TEMPLATE__ comment line. It contains the templated class's methods.

Use these macros in the same manner described for the TCL's templated classes. The TM_DECLARE_<classname> macro should appear ina header file with the regular templated class definition. The TM_DEFINE_<classname> macro should appear once per project per templated class instantiation. Use it in the same locations that you would use the #pragma template directive in the Symantec C++ compiler.

Template Munger has the following limitations:

- It will not process templated functions (such as the CStream helper functions in the TCL). You will need to write your own macros for such functions.
- Since it uses the preprocessor macros, it may be inefficient to use Template Munger for particularly large definitions. (In other words, I have not tested Template Munger with any classes which are more complex than the TCL templates.)
- The syntax parser in Template Munger is fairly rudimentary. If you find that it will not generate proper macros, try modifying the syntax.
- Template Munger merely discards all preprocessor macros (any line starting with a # is treated as a line comment). This essentially means that conditional compilations will evaluate to true in all cases (and that statements following an #else directive will also be included).

Exception Handling

The Bedrock Exception Library which was added to the TCL in version 2.0 seems to be somewhat buggy, especially on the PowerPC. For this reason, I have chosen to replace the BEL with the exception library which comes with PowerPlant on DR/3 and later. This change is triggered by a macro defined in the `CW HeadersTCL.h` file:

```
#if TCL_CW_VERSION>2
#define TCL_USE_PP_EXCEPTIONS
#endif
```

If you wish to retain the BEL in place of the PowerPlant library, you may comment out this macro.

You may continue to use all of the macros defined for the TCL without changing your source code, with the following restrictions:

- You may not use the typed exception handling which was added in the TCL 2.0. The macros `catch_`, `catch_reference_`, and `catch_no_instance_` are not permitted. You must use `catch_all_` instead.
- You may not reference the global variables `gAskFailure`, `gBreakFailure`, `_gTCLBreakCatch`, `_gTCLBreakFailure`, or `_gTCLBreakAssert`. These variables do not exist in this implementation. The macros which access these variables are implemented.
- The PowerPlant header <`UException.h`> must be installed in the CodeWarrior compiler folder. The file <`UException.cp`> must be added to the project.
- If the `__TCL_DEBUG__` macro is defined, the PowerPlant header <`UDebugging.h`> must also be installed in the CodeWarrior compiler folder. The file <`UDebugging.cp`> must be added to the project.
- Your source may not use the `throw_(exception)` macro. Error reporting must take place through the standard `Fail___` routines. The `throw_same_()` macro is permitted.

Note

The run-time type identification (RTTI) mechanism of the BEL is still used in the port package. ❖

Miscellaneous

Proposed Changes

This port package should be fairly useful as it stands, yet it is by no means complete. I hope to implement the following changes in the near future:

- Complete testing of Visual Architect classes and functions. (A diffs file for the VA builder may be necessary.)
- Incorporate bug fixes collected from the TCL 2.0 bug list.

Version History

Version 1.0, released 9 May 1994:

- Port package for CodeWarrior DR/2.

Version 1.2, released 26 May 1994:

- Updated for CodeWarrior DR/3.
- Replaced Bedrock exception library with PowerPlant exception library.
- Improved documentation.
- Skipped version 1.1 to avoid confusion with the TCL of the same name.

Version 1.2.1, released 27 May 1994:

- Quick fix to include a missing file (which wasn’t really missing, anyway).

Version 1.3, released 30 May 1994:

- Updated for TCL version 2.0.2.
- Included Template Munger application.

Acknowledgments

A special thanks to Jon Wätte who provided the MultiDiff application and inspired (challenged?) me to attempt this port, and to the several employees at Metrowerks who have supported this effort (especially Jonathon Hess).