## 1. CWEB Introduction.

The literate programming technique is described by Donald Knuth in "Literate Programming" and "The CWEB System for Structured Documentation". Literate programming adds new dimensions to a software developer's ability to document a program. The combination of a programming language with a typesetting language unleashes possibilities such as diagrams, mathematical formulas, and pretty printing. Hypertext links allow the association of source code with external documentation such as requirements, architectural design, functional specifications, or user guides. The organization of source code into sections encourages design by step-wise refinement.

An API documentation tool (as in javadoc or doxygen) is another key documentation tool which can be used in combination with literate programming. API documentation tools extract documentation for classes, class methods. class member data (object state), and function parameters.

The CWEB system depends on TEX for typesetting and Adobe PDF for display. Adobe Acrobat Reader and MiKTeX are available for free. CWEB source code is saved with a ".w" file extension. The CWEAVE program creates documentation from the CWEB source code by generating a TEX file. The TEX program generates a format called DVI that can be processed into postscript or PDF by executing "tex" and "dvipdfm". Alternatively, the PDF file can be generated in one step by executing "pdftex".

The build process for CWEB source code requires an extra step. The CTANGLE program generates the C++ source code for the C++ compiler. Microsoft Visual Studio allows customized build scripts. Assume a CWEB source file named classA.w_cpp has been added to a Visual Studio project. A customized build can be specified to integrate it into the normal build process for the project.

**2.** CWEB 3.64e Win32.

An installation package named cweb.exe installs the CWEB 3.64e Win32 version of CWEB on Microsoft Windows platforms. The distribution includes executable programs, user guides, examples, and source code. The CTANGLE program generates C++ source code. The CWEAVE program generates pretty printed documentation.

Hip, hip, hooray, a long-cherished dream is now a reality: CWEB version 3.6 introduced new features for clickable hypertext links by which you can read programs with Acrobat. It also contained extensive revisions that greatly improve its support for C++. And version 3.63 (January 1, 2001) is even better! My present intention is that version 3.63 will in fact be the very last upgrade (except of course for fixes to catastrophic bugs). -Donald Knuth

Professor Donald Knuth has given the software enginering community an excellent tool in CWEB. The tool is considered complete. There are a number of design decisions in the CWEB system that do not work well for a typical Microsoft Visual C++ project. This guide examines issues that may arise when using CWEB with Microsoft Visual C++. Revised versions of CTANGLE and CWEAVE plus some extra tools are included in CWEB 3.64e Win32. These changes facilitate the use of CWEB in the Microsoft Visual Studio C++ IDE.

First, the CTANGLE tool intentionally generates unreadable C++ files. Well formatted C++ files are required for a variety of reasons such as integration with other software development tools, viewing the source code in a format where the sections are expanded inline, and mixed team support (some team members will use literate programming and others won't). A number of modifications have been made in the CWEB 3.64e Win32 version of CTANGLE to generate readable C++ files via command line options.

Second, in CWEB all source files are named with the .w file extension. One style of CWEB programming is to generate multiple C++ files (usually a header file and source file for a class) from a single CWEB file. Another approach is to use one CWEB file to represent each C++ file. Using a one-to-one file mapping creates a familiar organization of source files and works well with incremental builds. The CWEB limitation is that there is no way to differentiate header and source files. In CWEB 3.64e Win32 the naming convention for the file extensions .w_h and .w_cpp are recognized by CTANGLE and CWEAVE.

Third, the CWEAVE tool considers all comments to be trailing comments. This poor assumption causes comments appearing before a line of code to be mistakenly associated with the previous line of code instead of the next line of code. The result is poor pretty printing. Software developers regularly use both forms of comment placement. Also, CWEAVE ignores any blank lines which separate code blocks. The CWEB 3.64e Win32 version of CWEAVE adds options to not assume that comment placement is trailing and to respect blank line dividers.

Fourth, the CWEAVE tool expects C++ comments to be in TEX format. Suppose existing C++ commentary is not written in TEX format. The CWEAVE process results in numerous TEX errors. Also, converting comments to TEX format breaks existing software development tools which expect comments to be in plain text. An alternative approach is to assume that comments are in plain text. TEX based C++ comments are identified using either *@ or \\@. This option is provided in the CWEB 3.64e Win32 version of CWEAVE.

Fifth, the CWEB system requires the starting position of unnamed sections to be marked with @@c. Suppose a software developer judges that CWEB commentary is not needed for a function because it already has reasonable commentary in both the header comment and inline comments. It is irritating to mark each such function with @@c. Also, by not inserting the @@c markers the CWEB source code looks like the equivalent C++. Only CWEB features such as named sections or CWEB commentary should require special marks. In the CWEB 3.64e Win32 there is an option to automatically insert @@c markers. See the file "preloader.cpp" for a description of the rules used for inserting unnamed section markers.

New CTANGLE command line options added in CWEB 3.64e Win32 are:
The option 'c' includes comments (default off).
The option 'd' includes section descriptions (default off).
The option 'l' includes line directive markers (default on).

The option 'm' includes section number markers (default on).
The option 'n' includes section name markers (default off).
The option 'r' sets the read only file attribute for output files (default off).
The option 'w' preserves indentation both spaces and tabs (default off).
The option 'y' automatically marks unnamed sections with @@c (default off)

New CWEAVE command line options added in CWEB 3.64e Win32 are:
The option 't' respects both regular and trailing comments (default off).
The option 'g' respects blank lines (default off).
The option 'y' automatically marks unnamed sections with @@c (default off)
The option 'z' indicates plain text comments (default off).

Additional programs are included in CWEB 3.64e Win32. 1) ACWEAVE executes the CWEAVE tool followed by the "pdftex" tool using a background process. 2) ACTANGLE executes the CTANGLE tool using a background process. 3) CWEBGEN converts existing C++ source code to CWEB source code and updates Microsoft Visual Studio project files. 4) CWEBSYNC operates recursively on all files in a directory. CWEBSYNC checks the synchronization of equivalent CWEB and C++ source files. CWEBSYNC copies files from the "expand" directory to the root directory. CWEBSYNC executes CWEAVE on all CWEB source files.

The CWEB 3.64e Win32 installer adds the CWEB executable directory to the Microsoft Visual Studio build environment. Confirm that the directory "C:\Program Files\CWEB\bin\" is registered as an executable directory. In Visual Studio 6.0 select "Tools (menu), Options (menu), Directories (tab), Executable Files (combo box)". In Visual Studio .NET select "Tools (menu), Options (menu), Projects (tree), VC++ Directories (tree), Executable Files (combo box)".

The CWEB 3.64e Win32 installer adds the CWEB file extensions to the list of file extensions using C++ syntax highlighting in the Microsoft Visual Studio editor .

For Visual Studio 6.0 add "w", "w_h", and "w_cpp" to the "FileExtensions" registry key.
HKEY_Current_User\Software\Microsoft\DevStudio\6.0\Text Editor\Tabs\Language Settings\C/C++

For Visual Studio .NET create registry key with default value set to editor GUID.
HKEY_Local_Machine\Software\Microsoft\VisualStudio\7.0\ Languages\File Extensions\.w
"{B2F072B0-ABC1-11D0-9D62-00C04FD9DFD9}"
HKEY_Local_Machine\Software\Microsoft\VisualStudio\7.0\ Languages\File Extensions\.w_h
"{B2F072B0-ABC1-11D0-9D62-00C04FD9DFD9}"
HKEY_Local_Machine\Software\Microsoft\VisualStudio\7.0\ Languages\File Extensions\.w_cpp
"{B2F072B0-ABC1-11D0-9D62-00C04FD9DFD9}"

**3. CWEB Custom Build Instructions.**

1) Name CWEB files using the file extension .w_h and .w_cpp for C++ header files and source files.

2) In a Microsoft Visual C++ project create two new folders named "CWEB Header" and "CWEB Source" to contain CWEB files. Select the project in the workspace view and then select Project, Add to Project, New Folder from the menu. The "CWEB Header" folder is associated with *.w_h files. The "CWEB Source" folder is associated with *.w_cpp files.

3) Add CWEB source files to the project. Select the project in the workspace view and then select Project, Add to Project, Files from the menu.

4) Customize the build process for CWEB source files. In Visual Studio 6.0 select the project in the workspace view and then select Project, Settings from the menu. In Visual Studio .NET select the project in the solution explorer, right-click, and select properties. In the "settings for" combo-box select "All Configurations" Select the "CWEB Header" folder. Select the "General" tab. Check "Always use custom build step". Select the "CWEB Source" folder. Select the "General" tab. Check "Always use custom build step".

5) Customize the release configuration. In the "settings for" combo-box select "Win32 Release".

Select the "CWEB Header" folder. Select the "Custom Build" tab.

In the "Commands" edit box type:
ctangle +ry "$(InputPath)"

In the "Outputs" edit box type:
$(InputDir)\$(InputName).h

Select the "CWEB Source" folder. Select the "Custom Build" tab.

In the "Commands" edit box type:
ctangle +ry "$(InputPath)"

In the "Outputs" edit box type:
$(InputDir)\$(InputName).cpp

6) Customize the debug configuration. In the "settings for" combo-box select "Win32 Debug".

Select the "CWEB Header" folder. Select the "Custom Build" tab.

In the "Commands" edit box type:
ctangle +ry "$(InputPath)"
actangle +cdnrwy -lm "$(InputPath)" nul "$(InputDir)\expand\$(InputName).h"
acweave +gtyz "$(InputPath)"

In the "Outputs" edit box type:
$(InputDir)\$(InputName).h
$(InputDir)\$(InputName)_h.pdf

Select the "CWEB Source" folder. Select the "Custom Build" tab.

In the "Commands" edit box type:
ctangle +ry "$(InputPath)"
actangle +cdnrwy -lm "$(InputPath)" nul "$(InputDir)\expand\$(InputName).cpp"
acweave +gtyz "$(InputPath)"

In the "Outputs" edit box type:
$(InputDir)\$(InputName).cpp
$(InputDir)\$(InputName)_cpp.pdf

7) Note: Define specific dependencies for each CWEB file if you are using CWEB change files, the CWEB include facility (@i file.w), or using metapost generated figures. Push the "Dependencies" button on the "Custom Build" tab and explicitly list all source file dependencies.

## 4. CWEB Tips.

The VIM editor knows CWEB syntax, see www.vim.org.

The CWEBINPUTS environment variable refers to the directory where CWEB searches for an included source file (if it is not found in the working directory). Store common CWEB files in the CWEBINPUTS directory. Check the content of the CWEBINPUTS environment variable in Windows 2000 using "Start Button, Control Panel, System, Advanced Tab, Environment Variables".

CWEAVE requires data types (including classes) to be indicated using the "@s datatype int" directive. A set of "@s datatype int" directives is available for common libraries such as the C++ standard library, MFC, and Win32 Platform SDK.

Add "@;" to the end of statements that are missing the closing ";" such as macros or sections. Comment out unused code using "#ifdef␣UNDEFINED" and "#endif" blocks. Do not comment out unreferenced parameters in functions, instead use "parName;␣//␣unreferenced␣parameter". Insert a forced line break using the "@#" or "@#@," directive. Reduce the indentation level using the "@t\4@>" directive.

Change files (".ch" extension) are used to patch the primary source file (".w" extension). Use a change file to revise CWEB for a specific operating environment. CWEB is UNIX oriented since it uses "dev\null" for an empty file. The Win32 version of the CWEB programs requires a change file to fix the "dev\null" lines. The same change file mechanism is used to extend or customize the functionality of CWEB.

Visual Studio .NET eliminated the MFC class wizard and distributed it's functionality to new places such as the class viewer. In Visual Studio .NET the wizard functions can only be applied to C++ files, so any changed changes must be copied to the CWEB source by hand.

PDF specific features can be used in CWEB source. Some useful PDF features are the ability to add graphics, web links using a URL, links within PDF documents, links to other PDF documents, annotations, and an outline index that displays in the right pane of Acrobat Reader. The pdfTex manual explains the available primitives for accessing PDF features. CWEAVE uses PDF features such as links through definitions in the macro package, cwebmac.tex. Use the macro \pdfURL{URL}{description} to add a URL link.

The METAPOST diagramming language is compatible with PDF through the \convertMPtoPDF macro. Execute METAPOST with the command "mp␣sourcefile". METAPOST produces encapsulated postscript files for each figure. For example the command "mp␣c:\texmf\doc\metapost\examples.mp" produces the files "examples.1" through "examples.9". At the beginning of your CWEB file add the statement "\input␣supp-pdf". Insert individual figures using \convertMPtoPDF{filename}{x scale}{y scale}.

A good set of code listings would consist of the documentation generated by CWEAVE, the interface documentation generated by an API tool, a listing of the CWEB source, and a listing of the expanded C++ source files. The cweave documentation can be generated on a file by file basis or as a module. Use Adobe Acrobat and Acrobat Distiller to combine the pieces (CWEAVE PDF, API documentation, CWEB source, and expanded C++ source) into a single PDF file if desired. Print using the 2 up (two pages on one page) option in landscape using duplex to save on paper costs.

Microsoft Word can be used to generate an effective listing. Keep your source code lines (web files) under 92 characters per line. A line greater than 92 characters may span multiple lines in Microsoft Word causing a mismatch in the line numbering. Copy and paste the source code into Microsoft Word. Select all. Set the font to "Courier New" with eight points. Set the tab spacing to 0.2 inches. Set the margins to 1.0 inches for the left and 0.5 inches for the right, top, and bottom. Add continuous line numbers (View, Page Layout followed by Page Setup, Layout Tab, Line Numbers). These settings can be saved in a Microsoft Word template for future use.

# Using CWEB with Microsoft Visual C++






Contributors: Frank A. Krueger