verifysoft
TECHNOLOGY

Tampere (Finland) / Offenburg (Germany), 4 July 2008

Please be informed that a new **CTC++ version 6.5.3** has been released.

Primarily a bug fix version. Also some enhancements: optimised multicondition coverage instrumentation, the CTC++ Preprocessor (ctc) component is now much faster, Microsoft C++/CLI extension "for each" is now handled, introduced Eclipse IDE integration on Windows, etc.

This file describes the changes in successive versions of CTC++.
The latest version is described first.

Version 6.5.3 (3 July 2008)

The previous version was v6.5.2. In it only the preprocessor (ctc[.exe]) component was new compared to v6.5. Since v6.5.2 there has been made available two unofficial/intermediate ctc[.exe] versions v6.5.3b and v6.5.3b2. The version comparison here is made to the v6.5.2 version.

Generally, this v6.5.3 is a bug-fix version (like v6.5.2), but now also some other components than ctc[.exe] have been touched.

This revision 6.5.3 of CTC++ has the following version numbers in its components:

Preprocessor	6.5.3	(was 6.5.2 seen with the -h option)
Run-time libraries	6.5.3	(was 6.5, seen by the 'ident' command applied on the library in some environments)
Postprocessor	6.5.3	(was 6.5, seen with the -h option and in the listings)
Header file ctc.h	6.5	(unchanged, seen in the ctc.h comments)
Configuration file ctc.ini	6.5	(unchanged, seen in the ctc.ini header)
CTC++ to HTML Converter	2.4	(was 2.3, seen with the -h option)
CTC++ to Excel Converter	1.1	(unchanged, seen with the -h option)
CTC++ Merger utility	1.0	(unchanged, seen with the -H option and in the merged listings)

and the following version numbers in its Windows platform specific components:

CTC++ IDE Integration	3.2	(was 3.1, seen by clicking the Tw-icon in the dialog program and selecting "About...". This integration is used at - Visual Studio .NET 2003/2005/2008 IDEs - CodeWarrior IDE [Symbian/emulator] - Carbide.c++ IDE [Symbian/emulator]) - Eclipse IDE
Visual Studio 5/6 Integration	2.2	(unchanged, version number seen by clicking the TW-icon in the CTC++ dialog boxes and selecting "About CTCui...")
CTC++ Wrapper for Windows	2.1	(was 2.0, seen by "ctcwrap -h")

and the following version numbers in its Unix platform (Linux, Solaris, HPUX) specific components:

CTC++ Wrapper for Unix	1.2	(unchanged, seen by "ctcwrap -h")

The corrections and enhancements in this version are the following:

In the CTC++ preprocessor (ctc):

- Problem fix: Some compilers (notably Visual C++) allow non-standard constructs like ...<::NameFromGlobalScope ...>... (A space should separate "<" and "::", as "<:" is a digraph for "[".) Such code may, however, occur in some commonly used header files (e.g., the Boost C++ libraries). Now this is handled by parsing "<::" as two tokens "<" and "::". This problem existed since v6.1.

- Bug fix: If there was correctly written "... < :: ...>", it could under certain special conditions become "... <:: ...>" in the instrumented code, and this does not compile with most compilers. Now fixed. This bug existed since v6.5.

- Bug fix: If there was conditional compilation within a template instantiation, like the following

```
SOME_TEMPLATE<...
#ifdef AAA
...
#else
...
#endif
...>...
```

the instrumented code became non-compilable (#line directives were embedded inside regular code lines). Now fixed. This bug existed since v6.5.

- Bug fix: The following kind of use case, a function returning pointer to function,

```
Rtype (*Fname(parlist1))(parlist2){...}
```

  was not recognized, and the function remained uninstrumented. Now fixed. This bug existed since v6.4.

- Bug fix: If the following kind of construct, "__pragma(arguments)" preceded a function definition, it could in certain cases happen that the function was not recognized and remained uninstrumented. Now fixed.

- Enhancement: In parallel builds, where the same symbolfile (e.g., MON.sym) is used, access to this file is serialised with a certain locking mechanism. An occasional inability to obtain the lock has been encountered in an LSF (Load Share Facility)/Linux Farm build environment where the symbolfile resided on a network drive. Now the algorithm is made more robust, and it seems to fix the problem. This problem existed since v6.3.

- Bug fix (Windows only): An UNC file name was not recognized as absolute. This could cause problems in these two cases.

  -- If the symbolfile was given as an UNC name, e.g.,

```
ctc ... -n \\servername\dirname\filename.sym ... cl ...
```

  it caused later a run-time error when running the instrumented program, because the datafile's name converted to an absolute form (from the symbolfile name) was erroneous.

  -- If the instrumented source file was given as an UNC name, e.g.,

```
ctc ... cl -c \\servername\dirname\sourcefile.cpp
```

  and the configuration parameter SOURCE_IDENTIFICATION was set to the value "absolute", the absolute name of the source file was erroneous in the symbolfile and in the listings.

- Bug fix (Windows only): When reinstrumenting a source file, the file name was given with different casing at the latter time, the file was considered to have changed even if it necessarily wasn't. This bug existed since v6.5.

- Enhancement: Optimised multicondition instrumentation. If the condition expression does not contain any && or || operators, its instrumentation is "reduced" to decision coverage, and the instrumentation overhead gets smaller. Also, the compiler error "operator && is ambiguous" (that could previously appear in some exceptional cases with C++) now disappears entirely from simple conditions like ...if (object)...

- Enhancement (Windows only): for each statements, a Microsoft C++/CLI extension in Visual C++ 8.0 (Visual Studio 2005) and later, are now allowed. Previously, CTC++ reported a syntax error. The for each statements, "for each(...in...){...}", are not instrumented, but they are shown in execution profile listings and other reports. So, usage of the for each construct does no more prevent from using CTC++ with the C++/CLI code (the /crl option).

- Enhancement: Speeded up the operation of ctc[.exe].

In the CTC++ run-time library:

- Enhancement: A similar enhancement as was made in the preprocessor with its symbolfile locking behavior (see above) is done also to the run-time library with its handling of parallel access to the same datafile (e.g., MON.dat). However, there have been no reports that parallel access would have been any problem here.

In the CTC++ postprocessor (ctcpost):

- Bug fix (Windows only): If there were two or more descriptions of the same source file with different timestamps (possibly instrumented independently from scratch), but yet representing the same level of the source file, ctcpost erroneously rejected the coverage data aggregation (summing), even if the only difference was in the casing of the source file names. This bug existed since v6.5.

- Enhancement: In connection with multiple symbol (or data) files containing descriptions (or coverage data) of the same source files, ctcpost displays certain informative notices how the second instance (description or coverage data) of the source file is treated. A couple of these notices removed in v6.4 have now been restored.

In CTC++ to HTML converter (ctc2html):

- Problem fix (Windows only): The directory parts of the source file names are normalised in a certain way (to deal with different casing and the optional ".\" notation). Now, there appears only one entry per each individual directory in the summary level HTML pages.

- Some other technical and cosmetic fixes.

In the CTC++ Wrapper for Windows (ctcwrap):

- Some changes needed for handling the new CTC++/Eclipse integration.

In the IDE integrations on Windows:

- Some enhancements and a bug fix in the Visual Studio .NET 2003/2005/2008 IDE integrations, read more from %ctchome%\Vs_integ\version.txt.

- Upgraded the CTC++ integration to work with Carbide.c++ v1.3. Previously only Carbide.c++ v1.2 was supported. Read more from %ctchome%\Sym_cw\version.txt

- Introduced IDE integration to Eclipse. Note that this is not a "plug-in" in the Eclipse sense. This is an arrangement, where the current dialog programs that are used in Visual Studio .NET 2003/2005/2008, Codewarrior, and Carbide.c++ IDE integrations are used also in Eclipse IDE. Read more from %ctchome%\Eclipse\readme.txt

Other Windows-specific issues:

- The installation program no more installs the license key (dongle) driver. We no more ship dongle-based licenses. Should a dongle driver be needed, it needs to be delivered and installed separately.

General:

- Enhancement: In heavy CTC++ use, and when the licensing arrangement is a floating license (by FLEXlm), it has sometimes happened that a license did not properly return to the pool of free licenses, not even when all linger and other timeout limits had expired. The license "hangs". Exact reason for this behavior is unknown. Now the license check-out/check-in arrangement has been changed so that the behavior should give minimally room for the "hung" licenses. Notably, the license manager no more needs to watch if the licensed program has been disconnected from the net or if it has possibly crashed.

- CTC++ User's Guide upgraded to v6.5.3 level.