Verifysoft TECHNOLOGY

Offenburg (Germany) / Tampere (Finland), 30 June 2016

Please be informed that a new **Testwell CTC++ version 8.0.1** has been released.

Testwell Oy — CTC++ System Version 8.0.1
Verifysoft Technology GmbH
30 June 2016

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

Version 8.0.1 (30 June 2016)

This revision 8.0.1 of CTC++ has the following version numbers in its components:

This revision has the following version numbers in its components:

Preprocessor	8.0.1	(was: 8.0; seen by -h option)
Run-time libraries	8.0.1	(was: 8.0; seen by 'ident' command applied on the library in some environments)
Postprocessor	8.0.1	(was: 8.0; seen by -h option and in the listings)
Header file ctc.h	8.0.1	(was: 8.0; seen in the file)
Configuration file ctc.ini	8.0.1	(was: 8.0; seen in the file)
CTC++ to HTML Converter	5.2	(was: 5.1; seen by -h option)
CTC++ to Excel Converter	3.3	(was: 3.2: seen by -h option)
CTC++ XML Merger utility	3.1	(was: 3.0; seen by -h option)
ctc2dat receiver utility	3.4	(was: 3.3; seen by -h option)

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

Visual Studio IDE Integration	4.2	(unchanged; seen by clicking the Tw-icon in the dialog program and selecting "About...")
CTC++ Wrapper for Windows	3.4	(unchanged: seen by -h option)

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

CTC++ Wrapper for Unix	1.4	(unchanged; seen by -h option)

In the CTC++ preprocessor (ctc):

- Bug fix: Relational operator < was a problem to ctc in template instantiations and specialisations, and could in extreme cases cause unpredictable consequences, e.g.,
  ```
  template <bool b> class X { ... }; ... X< A<B > ...
  ```

- Bug fix: A lambda following immediately assignment operator '=' was not instrumented, when instrumentation mode was multicondition coverage, e.g.,
  ```
  i = [](){ return 0; };
  ```

- Bug fix: A lambda in global scope was not instrumented, if it was assigned to a function pointer, e.g.,
  ```
  int& (*fpi)(int*) = [](auto* a) -> auto& { return *a; };
  ```

- Bug fix: A function or lambda was not instrumented, if there was '&&' (C++11) in its trailing return type, e.g,,
  ```
  []() -> int && {...}
  ```

- Bug fix: When in newer C++ it came allowed to write initialization without '=', e.g. 'int buf[100]{0};', ctc no more (erroneously) instruments the code like a lambda.

- Bug fix: In some connections, e.g. as a function parameter, ctc reported wrongly syntax error of a lambda function that had '[=]' capture when instrumentation mode was multicondition coverage, e.g.
  ```
  f( [=]()->int{ return 1; }() );
  ```

- New: Some new C++14 features are now properly handled:
  -- binary literals, e.g., 0b010100
  -- digit separator in numeric literals, e.g., 109'208'000
  -- 'auto' in a trailing return type and in a parameter list

- Bug fix: String literal like "AAAxBBB" where x was binary 1 was not parsed properly.

- Enhancement: Certain kind of inline assembly functions, supported by some compilers, are now handled (recognized, not instrumented, body lines kept unchanged) automatically. Such functions look like
  ```
  [_][_]asm[_][_] rtype fname(plist){assembly lines}
  ```
  Previously RUN_AFTER_CPP/RUN_AFTER_INSTR extra scripts were used to handle these cases.

- Enhancement: On primary host platforms (cl and gcc/g++ compilers) fine-tuned the default ctc.ini files to work more properly with some newly encountered option usages.

- Bug fix: A 'return expression;' statement was not instrumented correctly and resulted in a compilation error, if there was '#include "otherfile"' immediately after 'return', e.g.,
  ```
  return
  #include "otherfile"
  ;
  ```

- Change/bug fix: Now, if ternary-? appears in a 'typedef ...;' construct, it is never instrumented. Instrumentation could cause a compile error with some compilers.

- Bug fix: Initializations having 'const' or 'constexpr' qualifier, e.g. 'const bool b = aa && bb;', are no longer instrumented. Compile error would occur if variable 'b' were later used in a place needing the 'const' or 'constexpr' property.

- Bug fix: If there was a 'friend' function in a class and the function had also a body, the function remained uninstrumented. Now it is instrumented. E.g.,
  ```
  class X { ... friend int fr1(...){...} ... };
  ```

- Bug fix: GCC's 'asm goto (...);' and 'asm volatile goto (...);' are now handled correctly. Previously, they were instrumented like normal goto statements (yielding non-compilable code).

- Enhancement: In the configuration parameter EMBED_FUNCTION_NAME, wildcard character(s) '*' are now allowed. E.g.,
  ```
  -C "EMBED_FUNCTION_NAME=*::close"
  ```

In the CTC++ run-time library:

- Change: In ctc.h, at Windows (when 'defined(_WIN32)') the default storage class specifier on the CTC++ run-time functions, ctc_register_module() etc., is now simply 'extern'. Was previously '__declspec(dllimport)'. Now default instrumentation of the code files and plain compile of the HOTA targ*.c files link smoothly. Previously there was problems e.g. with gcc. Should the HOTA-based runtime be a static/dynamic library and need special linker advice, there are means to introduce them when instrumenting the code and when compiling the HOTA targ*.c files.

- Change: At Linux platform started to use flock() system service to additionally ensure exclusiveness in datafile handling (was needed in some extreme situations).

In the CTC++ postprocessor (ctcpost):

- Bug fix: no more crash (in some cases) when two reporting options in the command e.g. -p and -u. (in future this may become denied...)

- Bug fix: E.g. the following 'if (c1 ? 1 : 0) {...' gave wrong statement coverage result, when c1 was evaluated but always to false. Silly code as such, but possible.

- Bug fix: When handling header file names, and if they contained duplicated directory separators \\ or // in some special cases.

- New: In .txt and .xml profile report there is new bottom line summary information "Functions : n". It tells how many instrumented functions there are in the reported files altogether. In .xml report, the number of functions is additionally reported per each file.

In the Visual Studio IDE Integration:

- Bug fix: The "basic integration engine" (vsCTC.exe) is unchanged, but at installation time there is improved modify_msbuild_path.bat, which enables C# instrumentation in Visual Studio 2013 and later.

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

- Documentation fix: The previous v8.0 version.txt said ctcwrap still be subversion v3.3, while it was already subversion v3.4.

In CTC++ to HTML converter (ctc2html):

- New: Advanced option --enable-statement_coverage=0/1 added. With it statement coverage information can be dropped off from the HTML report (becomes simpler, has less "eye-stoppers").

- Enhancement: New heuristic to conclude if a function internal conditional code block (#if...#elif...#else...#endif) is in the build or not, and determine the line coverage background color accordingly. Now if such block has no counters, it is assumed not to be in the build (grey letters on white background in the HTML). Otherwise the block is painted on the line coverage color (green/red) as concluded by the function execution flow analysis. This heuristic is not 100% correct, but more correct than before.

- Bug fix: A used -s option now shows properly in the HTML report at various directory fields. Example: Assume a file has been instrumented so that it is known in Execution Profile Listing as '..\Dir\file.c'. Then, ctc2html is run in a directory where that name does not resolve. Assume the file really resides at 'F:\Work\Dir\file.c' and with option '-s F:\Work\Dir' ctc2html can find the file (and html'ize it). Now in the HTML report the file's directory is 'F:\Work\Dir', was '..\Dir'.

- Bug fix: When in a string literal there was something looking like a comment start, or the other way round.

- Changes/Bug fixes: Fine-tunings or small corrections how the line coverage background color gets determined in various special cases.

In the CTC++ XML Merger utility:

- Enhancement: ctcxmlmerge is now made much faster.

- Change: The report now shows correctly copyright year (reflecting when the ctcxmlmerge utility was made), no more the copyright year of the first input XML report (reflecting the year when the used ctcpost utility was made).

In the CTC++ to Excel Converter (ctc2excel):

- Enhancement: Now properly handling input Execution Profile Listings
  that are generated by ctcpost v8.0.1 (having the number of functions
  at the bottom summary).

General:

- CTC++ User's Guide upgraded to v8.0.1 level (ctcug.pdf).

- The 'stack' example changed so that it no more uses uninitialized
  variables. Some debuggers noticed it and stopped example running.

Version 8.0 (27 November 2015)

For this version, please have a look to
http://verifysoft.com/ctcpp80.pdf