16 December 2022

Change Documentation for

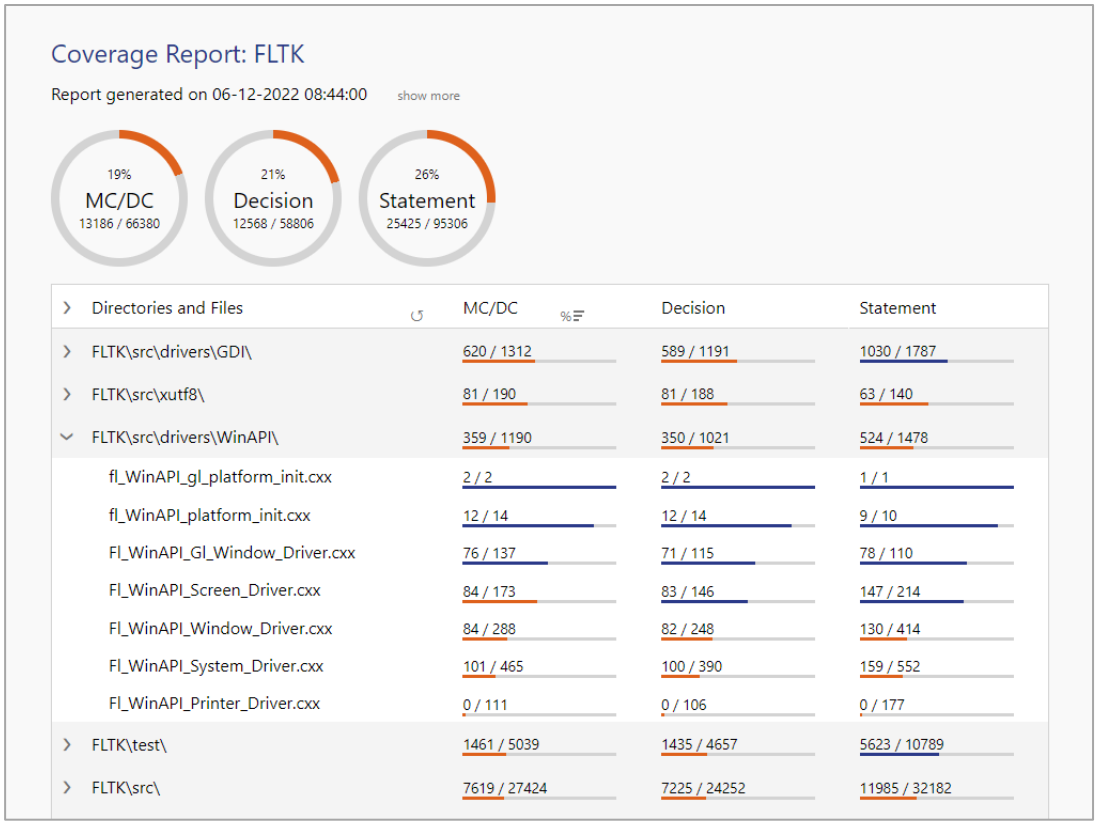
Testwell CTC++

Version 10.0.0

Features and Changes

New Generation of HTML Reports

With version 10, we introduce the new tool **ctcreport** to generate HTML reports directly from symbol and data files. A detailed feature description is available in the Testwell CTC++ Help. In this document, improvements and changes compared to the former report generation are described.



Adaptable Layout

You can choose a layout fitting to the structure of your project. A layout determines the levels of detail (directories, files, functions) reported and the kind of HTML pages (overview page, detail pages, source code views) generated. Layouts are configured in **ctcreport-layout.ini** in the ctcreport-folder.

Reducing Generation Steps

No intermediate text report (**profile.txt**) is needed anymore for generating an HTML report.
The report generation is generally also faster compared with a **ctcpost** + **ctc2html** call.

Combining all Coverage Measures

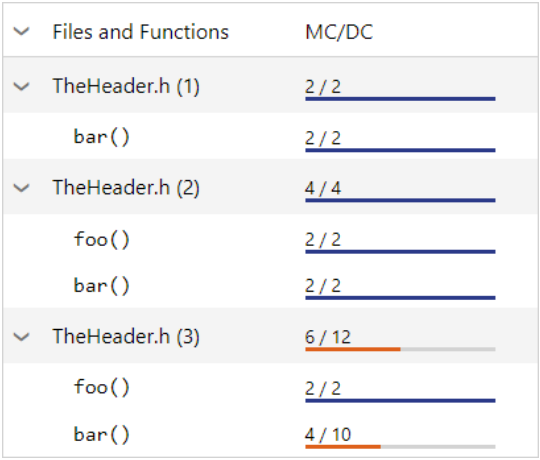
You can choose any combination and the order of coverage measures to be shown in the report. With option `-measures mcdc,d,s` the report in the screenshot above was generated.

Project Name

You can assign the name of the project with option `-D "ProjectName=My Super Tool"`. The project name is shown in the caption of the report.

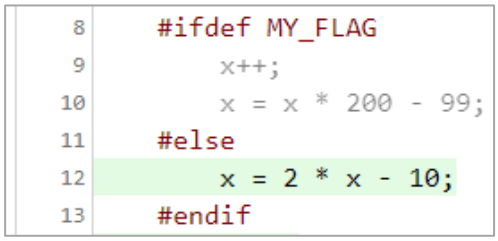
Reporting for Header Variants

When an included file is preprocessed in more than one variant, **ctcpost** only “extracts” the first variant found and reports all other variants as parts of their including files.
In contrast, **ctcreport** reports every variant of a header file as an independent entity. Included files are never reported as part of their including files.



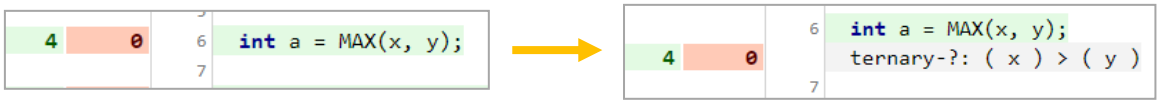
Identification of Conditionally Compiled Code

Code between conditional preprocessor directives is now identified as active or inactive in all cases, not only if there is an instrumentation probe from **ctc** inside. Hence line coverage and the visual indication that code parts have been inactivated are reliably derived by **ctcreport**:



Displaying all Probe Information

In the common case with one instrumentation probe on a source code line, the source code view shows the true and false counters next to the line and suppresses the probe type and description. When macros are used, for example, this can lead to counters with an unclear origin.
In the new HTML report, you can show all probes by selecting *Source & Details > Show Coverage Details > Single Probe Descriptions*.



With more than one probe on a line, the first probe description is now always shown (as all the other probes on the same line).

Assisting Functions inside the Report

In all tabular views, you can sort the table data by coverage ratio or by the number of missing hits for each coverage measure.
In the source code view, you can choose the details to be shown and you can highlight the counters for a certain measure to understand the calculation.

Templated HTML

The HTML code generated is completely drawn from an HTML template set called **html_template.zip** located in the ctcreport-folder. This package contains templated HTML files for the overview page, detail pages and for source code views as well as CSS and JavaScript resources.
It is possible to adapt the template. In this case, make a copy of the original one, archive it to a ZIP package like **my_template.zip** after changing, and call **ctcreport** with option `-template my_template`.

Handling of different build / reporting directories

When the source code is located in different directories during instrumentation compared to reporting time, a combination of SOURCE_IDENTIFICATION set to something different than "absolute" and the option `-s` of **ctc2html** to find source files in different folders was used in the past. This functionality is completely replaced by a new function: Mapping source identifications recorded in symbol files to source paths available for reporting. The option `-map-source-identification` is used for that purpose.
For a convenient display of long paths, the option `-shorten-path` is available. For that purpose, there is no need to use `SOURCE_IDENTIFICATION = as_given` anymore.

Changes in the delivery package

In consequence, **ctc2html** is no longer part of Testwell CTC++. **ctcpost** is still used for generating text or XML reports and for analyzing and combining symbol and data files.

Note on "Merging"

The workflow

1. Generate XML reports,
2. Merge them with **ctcxmlmerge** to a text report,
3. Generate an HTML report from that text report with **ctc2html**,

is not fully supported anymore. In most situations, you can combine symbol and data files to an HTML report instead.
For source files preprocessed in different variants (due to `#if`, for example), **ctcreport** shows in one report every variant of that source or header file independently but does not merge coverage data from different variants.

The build launcher **ctclaunch** for Linux

The tool **ctclaunch** can be used to interact directly with build tools like CMake, taking the build command as input and taking care that **ctc** is invoked with all compiler and linker calls.
ctclaunch has been available since version 9.1 for Windows and is now introduced for Linux.

We recommend using **ctclaunch** instead of **ctcwrap** (standard or in "-hard" mode) whenever possible. Please get in contact with your support team if you face any issues with that replacement. We expect to withdraw **ctcwrap** in a future release.

Changes with **ctclaunch** configuration on Windows

To streamline the use of **ctclaunch** on Windows and Linux, a configuration file called **ctclaunch.ini** is used. All compilers and linkers used during build must been listed in the new configuration parameter PREPEND_CTC in this file, located in CTCHOME.

```
PREPEND_CTC = cl, link
PREPEND_CTC + gcc, g++, ld
```

The file **wrapcmds.txt** is no longer used by **ctclaunch**. With an upgrade, you must check the compiler settings in **ctclaunch.ini** once.

Deactivate instrumentation for missing defaults

Since version 9.1, missing defaults of switch-case statements are instrumented and must be tested. For C++ code, this could cause compilation issues with initializations in last case. This code

```
switch (x)
{
    case 1:
        do_something();
    case 2:
        int a = 666;
        do_something();
}
```

can be instrumented when compiled as C code, but not when compiled as C++ Code.
With version 10, a new configuration parameter MISSING_DEFAULT_INSTR is introduced and set to ON in **ctc.ini** files delivered.
Set

```
MISSING_DEFAULT_INSTR = OFF
```

to prevent the instrumentation.

Symbol and Data Files

As this is a major release, you must generate symbol and data files from scratch. They cannot be used from former releases.

Reporting for Skipped Code

Using #pragma CTC SKIP / #pragma CTC ENDSKIP and their string variants "CTC SKIP" / "CTC ENDSKIP" has now the same consequences regarding warnings, calculation of statement coverage and highlighting of lines:
As far as possible, both measures are derived even if the #pragma variant is used inside a function. Both measures can be misleading if the structural flow is affected, hence a warning is issued by **ctc** for all cases.
When using the pragma variant inside a function, an additional `#pragma CTC COUNT name` can be used immediately after `#pragma CTC ENDSKIP` to ensure that statement and line coverage are derived correctly after this point again.

Windows: Signing of Executables

On Windows, the installer and all executables are now signed with a code signing certificate. Besides, an MD5 hash is available for all ZIP packages.

Licensing

Starting with version 10, license control includes the release date of the Testwell CTC++ version used. If you have an older license file, please contact your sales manager for a new one. New license files allow using versions released until the end of your maintenance period.

Bug Fixes

Const-recognition

The extended recognition of compile-constant decisions in `if`-statements etc. introduced with version 9.0.0 could lead to crashes and wrong categorization. This recognition is now replaced by a new implementation.
Now a decision is recognized as compile-time constant if it consists only of strings, numbers, characters, brackets, operators, `sizeof` operator, standard C types, const-modifier `const`.