
# Release Notes

## UCC v.2011.10

# 1 Introduction

This document provides the release notes for the UCC v.2011.10. Unified CodeCount (UCC) is a code counting and differencing tool that allows the user to count physical and logical software lines of code, and compare and collect logical differentials between two versions of source code of a software product. With the counting capabilities, users can generate the physical, logical SLOC counts, and other sizing information such as comment and keyword counts of the target program. The differencing capabilities allow users to count the number of added/new, deleted, modified, and unmodified logical SLOC of the current version in comparison with the previous version.

This release supports both counting and differencing for various languages including Ada, ASP/ASP.NET, Bash, C/C++, C Shell, ColdFusion, ColdFusion Script, CSS, C#, Fortran, HTML, Java, JavaScript, JSP, NeXtMidas, Pascal, Perl, PhP, Python, Ruby, SQL, VB, VBScript, XML, and X-Midas. It also supports physical counting of data files.

# 2 Compatibility Notes

UCC v.2011.10 is released in C++ source code which allows users to compile and run on various platforms. This release has been tested on Windows using MS Visual Studio, Cygwin, and MinGW, and on Unix/Linux using the g++ compiler.

The UCC v.2011.10 does not support Assembly, PL/1, COBOL, and Jovial, although these may be included in future releases. For the need of counting of code in these languages, users may consider using the CodeCount Tools Release 2007.07 which does not provide the differencing capability but uses the counting rules compatible to those of UCC v.2011.10.

# 3 Requirements

Minimum Software Requirements:

- Compiler: a compatible C++ compiler that can load common C++ libraries including IO and STL, such as MS Visual Studio, MinGW, and g++.
- Operating systems: any platforms that can compile and run a C++ application. The software has been tested on Windows XP/7, Unix, Linux, Solaris, and Mac OS X.

Minimum Hardware Requirements:

- RAM: 512 MB. Recommended: 1024 MB.
- HDD: 100 MB available. Recommended: 200 MB available.

# 4 Features

1) Counting Capabilities. UCC allows users to measure the size information of a baseline a source program by analyzing and producing the count for:
   - logical SLOC
   - physical SLOC
   - comments, whole and embedded
   - executable, data declaration, compiler directive SLOC
   - keywords
   - complexity measures: mathematic functions, logarithms, calculations, assignments

2) Differencing Capabilities. UCC allows users to compare and measure the differences between two baselines of source programs. These differences are measured in terms of the number of logical SLOC added/new, deleted, modified, and unmodified.

3) Reports. A variety of reports are produced. The default report format is .csv, which will open directly into Excel, but plain text reports with the extension.txt can be specified by using the –ascii switch.

4) Counting and Differencing Directories. UCC allows users to count or compare source files by specifying the directories where the files are located. This capability eliminates difficulties in creating the file list that users may have encountered in the previous versions of the CodeCount toolset.

5) Various Programming Languages Supported. The counting and differencing capabilities accept the source code written in Ada, ASP/ASP.NET, Bash, C/C++, C Shell, ColdFusion, ColdFusion Script, CSS, C#, Fortran, HTML, Java, JavaScript, JSP, NeXtMidas, Pascal, Perl, PhP, Python, Ruby, SQL, VB, VBScript, XML, and X-Midas. The tool detects the language of each file using its file extension (see Feature #11)

6) Command Arguments. The environment file containing user's settings (e.g., c_env.dat file) in the CodeCount tools is no longer used. Instead, the tool accepts user's settings via command arguments. Specifics of the command arguments are detailed in the UCC User's Manual.

7) Duplication. For each baseline, two files are considered duplicates if they have same content or the difference is smaller than the threshold given through the command line switch -tdup. Two files may be identified as duplicates although they have different filenames.

For counting, duplicates in the input files are counted and their counting results are saved into a file named Duplicates-<LANG>_outfile.csv, where <LANG> is the name of the programming language used. Duplicate file pairs are identified in a file named DuplicatePairs.csv, with matching pairs displayed in two columns. The complexity metrics of the duplicate files are reported in a file named Duplicates-outfile_cplx.csv.

For differencing, duplicates in each baseline are counted, and their counting results are saved into files named "Duplicates-A-<LANG>-outfile.csv" and "Duplicates-B-<LANG>-outfile.csv". As such, one or more files are generated as a result of the duplication feature. Duplicate pairs are identified in files Duplicates-A-DuplicatePairs.csv and Duplicates-B-DuplicatePairs.csv. The complexity metrics of the duplicate pairs are reported in a file named Duplicates-A-outfile_cplx.csv and Duplicates-B-outfile_cplx.csv. Note that duplicates are identified within baselines, and not across baselines.

Comments and blank lines are not considered during duplication processing for files which have the same filenames.

8) Duplication Matching. Two files are matched if they have the same filename regardless of which directories they belong to. Two files that have the same filename are matched if they have the least uncommon characters in their directory names. This feature allows users to handle to the situation where files are moved from one directory to another or the directory structure is changed. The remaining files are matched according to an algorithm that makes the most likely match.

9) Complexity Count. UCC produces complexity counts for all source code files. The complexity counts include the number of math, trig, logarithm functions, calculations, conditionals, logicals, preprocessors, assignments, and pointers. When counting, the complexity results are saved to the file "outfile_cplx.csv", and when differencing the results are saved to the files "Baseline-A-outfile_cplx.csv" and "Baseline-B-outfile_cplx.csv". Note that the cyclomatic complexity counter is being updated for a future release and is not currently supported.

10) Unix/Linux. Under Unix/Linux when using the –dir option, any wildcards must be enclosed within quotes. Otherwise, the wildcards will be expanded on the command line and erroneous results will be produced. For example: ucc –d –dir baseA baseB *.cpp should be written as ucc –d –dir baseA baseB "*.cpp".

11) File Extensions. The tool determines the language used in a source file using file extension. This release supports the following languages and file extensions:

- ADA, ASP/ASP.NET, Bash, C/C++, C Shell, ColdFusion, ColdFusion Script, CSS, C#, Fortran, HTML, Java, JavaScript, JSP, NeXtMidas, Pascal, Perl, PhP, Python, Ruby, SQL, VB, VBScript, XML, and X-Midas

| Languages | File Extensions |
|---|---|
| **Ada** | .ada, .a, .adb, .ads |
| **ASP, ASP.NET** | .asp, .aspx |
| **Bash** | .sh, .ksh |
| **C Shell Script** | .csh, .tcsh |
| **C#** | .cs |
| **C/C++** | .cpp, .c, .h, .hpp, .cc, .hh |
| **ColdFusion** | *.cfm, .cfml, .cfc |
| **ColdFusion Script** | .cfs |
| **CSS** | .css |
| **Data** | Use file mapping with Datafile=<ext> |
| **Fortran** | .f, .for, .f77, .f90, .f95, .f03, .hpf |
| **HTML** | .htm, .html, .shtml, .stm, .sht, .oth, .xhtml |
| **Java** | .java |
| **JavaScript** | .js |
| **JSP** | .jsp |
| **NeXtMidas** | .mm |
| **Pascal** | .pas, .p, .pp, .pa3, .pa4, .pa5 |
| **Perl** | .pl, .pm |
| **PhP** | .php |
| **Python** | .py |
| **Ruby** | .rb |
| **SQL** | .sql |
| **VB** | .vb, .frm, .mod, .cls, .bas |
| **VBScript** | .vbs |
| **X-Midas** | .txt |
| **XML** | .xml |

# 5 Changes and Upgrades

This section describes changes and upgrades to the tool since the release v.2011.03.

1) Bug Fixes:

   a. Updated all counters to ensure consistent counts within the counting and differencing reports
   b. Updated all comment/quote handlers to correctly and consistently process blank lines and comment types
   c. Reduced formatting dependencies by removing redundant spaces in a line and line continuation characters
   d. Fixed file listing to correctly handle spaces in file path names
   e. Added trapping of bad file streams
   f. Fixed C# bugs counting verbatim string (@"\") and declarations with casts
   g. Fixed CSS counter algorithm
   h. Fixed Fortran bugs handling embedded quotes and F77/F90 differences
   i. Added regular expression handling in JavaScript
   j. Added handling of Perl expressions modifiers and variable placeholders ($', $", $`)
   k. Fixed handling of multi-line literals and comments using """ and "' in Python
   l. Updated SQL counter to process by line to allow for large files
   m. Fixed –extfile bug where HTML is skipped if XML extensions are specified
   n. Added filtering of Windows smart quotes
   o. Added –nolinks option to skip Unix symbolic links (prevents multiple counting of same files as links)
   p. Fixed some quote processing issues
   q. Fixed C# verbatim string processing
   r. Fixed a variety of Ruby defects
   s. Updated the Ruby counting standard document

2) New Features:

   a. Added processing of directories in the fileList.txt files
   b. Added embedded, whole-line, and multi-line comments to the –extfile (comments are enclosed in square brackets [ ])
   c. Added usage help for each command-line option (UCC –h -<option>)
   d. Added compiler support for MinGW compiler for Windows (-DMINGW compile option)
   e. Added executed command line to report headers

3) New Languages

   a. Data – physical counting only. The user must specify an extension file using the –extfile <filename> switch, and the contents of the file must include Datafile=<.ext>. See the User's Manual for more information.
   b. Separated the XML counter from HTML and the VBScript counter from VB to allow for user specification of extensions.

4) New Reports:
   a. Added a language count summary report outfile_summary.csv
   b. Added an uncounted files report outfile_uncounted_files.csv to list all files encountered which could not be processed along with the reason (if known) why it could not be counted.

# 6 Known Issues and Limitations

| # | Issue |
|---|---|
| **1** | For JavaScript code, the tool does not count the statement that is not terminated by a semicolon. |
| **2** | The tool only detects and handles C# and VB as code-behind languages for the ASP.NET. |
| **3** | Users have reported that when large numbers of files or files with large SLOC counts are run, UCC would take several hours to process, or would hang. To improve the performance, users may choose to use the –nodup flag, which disables duplicate file separation; duplicates are counted and reported along with original files. In the situation where UCC hangs, the problem is that the host computer has run out of memory. A workaround is to break the input file list into several lists and process in multiple runs. Additional work is being done in this area, and more improvements may be available in the next release.<br><br>If you suspect your process is hanging due to memory limitations, it would be appreciated it if you would report the number of files, total file size, and the host computer's memory size to UnifiedCodeCount@gmail.com. |
| **4** | The UCC is designed to process well formed, compilable code, and does not check to see if the provided files are compilable. Files that contain software that is not compilable, or is in non-standard format, may not process correctly. |
| **5** | The Fortran counter uses the FORTRAN90 and above format for the continuation character being an & at the end of the line. FORTRAN77 and lower versions used a non-zero character in column 6 as the continuation character. There are plans to develop a separate counter for FORTRAN77 and lower in the future. |