Conversion Tools - Bug #3242

parser console output written out of order

02/07/2017 11:01 AM - Eric Faulhaber

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eric Faulhaber	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 02/07/2017 11:26 AM - Eric Faulhaber

The parsing step of the front end conversion phase uses a thread pool managed by an ExecutorService (see ScanDriver.scan) to allow concurrent parsing of files. A side effect is that messages written to the console can appear out of order, and always appear before the console message which prints the name of the file being parsed, because the initial file name is printed from the main thread when processing execution results, while any other messages are printed from independent threads during parsing. As a result, the association between a message and the file it affects is lost or confused.

Multi-threaded parsing is useful to cut down conversion time on large projects that are mature and have had all initial parsing problems worked out. However, using a single thread for parsing is the ConversionDriver default, because new projects typically will have parsing errors and warnings, and it is easier to track these down when the message corresponds with the file in which the problem occurred. Even when a single thread is specified, though, messages appear before the file name is reported, which is confusing.

The point of this task is to bypass the executor/thread pool approach when a single parsing thread is specified, so that error and warning messages appear directly after the file name being parsed. The executor should be used only when multiple threads are specified.

#2 - 02/14/2017 10:29 AM - Eric Faulhaber

- % Done changed from 0 to 80

- Status changed from New to Test

A fix is committed to 3209e/11208. We do not bypass the thread executor service as proposed above, but we use it with a single thread and force the filename to be printed before scanning the file in this use case (the default). This effectively matches the behavior of the original, single-threaded implementation.

Note that when run with multiple threads, the filenames still are printed after the files are processed. This is necessary (a) to preserve the order in which they were submitted; and (b) to prevent all the filenames from being dumped to stdout immediately (i.e., before they are processed) in the main thread.

We could print the filenames in the worker lambda just before they are parsed. In fact, this is what we are doing in the single thread case. However, this doesn't work well for the multi-threaded case, because it does not preserve the original order of the files. This makes comparing conversion logs from different runs problematic, as the same converted result across two runs can produce very different logs, even when the conversion outcome is otherwise identical.

If we changed this to print the filenames in the main thread as they are submitted to the executor service, there would be a burst of filenames sent to stdout, followed by a long wait with no feedback as the queue of files was processed. This would be confusing to anyone monitoring the output in real

time.

#3 - 03/14/2017 01:57 PM - Greg Shah

- % Done changed from 80 to 100

- Status changed from Test to Closed

Additional changes were applied by ECF in branch 3255a, which was merged to trunk as revision 11143. At this time, all error output is done on a single thread and the order of the error output should be the same as before the executor framework was introduced.

Some out of order output can still be seen, but this may have been present originally and/or it is related to how the STDOUT/STDERR gets used/flushed.