



INTEL[®] INSPECTOR

Thread & Memory Debugger

Klaus-Dieter Oertel
Intel IAGS

HLRN User Workshop, 3-6 Nov 2020

Race Conditions Are Difficult to Diagnose

They only occur occasionally and are difficult to reproduce

Correct

Thread 1	Thread 2		Shared Counter
			0
Read count		←	0
Increment			0
Write count		→	1
	Read count	←	1
	Increment		1
	Write count	→	2

Incorrect

Thread 1	Thread 2		Shared Counter
			0
Read count		←	0
	Read count	←	0
Increment			0
	Increment		0
Write count		→	1
	Write count	→	1

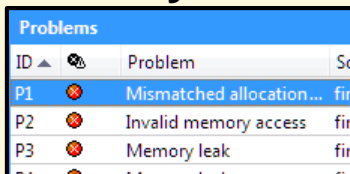
Deliver More Reliable Applications

Intel® Inspector and Intel® Compiler

Intel® Inspector

- Dynamic instrumentation
- No special builds
- Any compiler¹
- Source not required

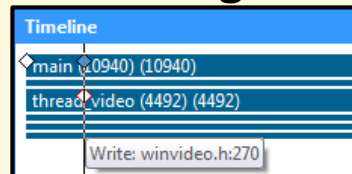
Memory Errors



ID	Problem	Source
P1	Mismatched allocation...	fin
P2	Invalid memory access	fin
P3	Memory leak	fin

- Invalid Accesses
- Memory Leaks
- Uninit. Memory Accesses

Threading Errors

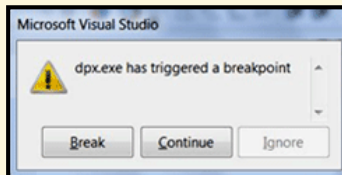


- Races
- Deadlocks
- Cross Stack References

Intel® Compiler

- Pointer checker
- Run time checks
- C, C++

Pointer Errors



- Out of bounds accesses
- Dangling pointers

Find errors earlier with less effort

¹That follows common OS standards.

Productive User Interface Saves Time

Intel® Inspector

Select a problem set

Code snippets displayed for selected problem

The screenshot displays the Intel Inspector 2017 interface. At the top, the title bar reads "Detect Memory Problems" and "INTEL INSPECTOR 2017". Below the title bar, there are tabs for "Target", "Analysis Type", "Collection Log", and "Summary". The main area is divided into two panes. The left pane, titled "Problems", contains a table of detected issues. The right pane, titled "Filters", allows for filtering problems by severity, type, and source. Below these panes, a "Code Locations" pane shows the source code for a selected problem, with specific lines highlighted in yellow.

Type	Sources	State	M
Mismatched allocation/deallocation	find_and_fix_memory_...	Confirmed	
Memory leak	find_and_fix_memory_...	Deferred	fi.
Invalid memory access	find_and_fix_memory_...	New	fi.
Memory not deallocated	api.cpp; mlock.c; util.c...	New	fi.
Memory not deallocated	video.cpp:82	New	fi.
Memory not deallocated	util.cpp:163	New	fi.
Memory not deallocated	api.cpp:218	New	fi.
Memory not deallocated	mlock.c:347	New	tb.

Severity	Count
Error	3 item(s)
Warning	1 item(s)
Invalid memory access	1 item(s)
Memory leak	1 item(s)
Memory not deallocated	1 item(s)
Mismatched allocation/deallocation	1 item(s)

Description	Source	Function	Module	Object Size	Offset
Mismatched deallocation site	find_and_fix_memory_errors.cpp:175	operator()	find_and_fix_memory_errors.exe		
173	drawing->put_pixel(c);				
174	}				
175	free(drawing); //Memory Error: use delete instead of free				
176	//delete drawing;				
177	}				
Allocation site	find_and_fix_memory_errors.cpp:170	operator()	find_and_fix_memory_errors.exe		
168	for (int y = r.begin(); y != r.end(); ++y) {				
169	{				
170	drawing_area * drawing = new drawing_area(startx, totaly-y, st				
171	for (int x = startx; x < stopx; x++) {				
172	color_t c = render_one_pixel(x, y, local_mbox, serial, st				

Filters let you focus on a module, or error type, or just the new errors or...

Problem States: New, Not Fixed, Fixed, Confirmed, Deferred, Not a problem, Regression

Double Click for Source & Call Stack

Intel® Inspector

Source code locations displayed for selected problem

Mismatched allocation/deallocation

INTEL INSPECTOR 2017

Target Analysis Type Collection Log Summary Sources

Mismatched deallocation site - Thread thread_video (4596) (find_and_fix_memory_errors.exe!operator()) - find_and_fix_memory_errors.cpp:170

find_and_fix_memory_errors.cpp Disassembly (find_and_fix_memory_errors.exe!0x46d6) Call Stack

```
164
165     for (unsigned int i=0;i<=(mboxsize/(sizeof(unsigned int)));i++)
166         local_mbox[i]=0; //Memory Error: C declared arrays go from 0
167
168     for (int y = r.begin(); y != r.end(); ++y) {
169         {
170             drawing_area * drawing = new drawing_area(startx, totaly
171             for (int x = startx ; x < stopx; x++) {
172                 color_t c = render_one_pixel (x, y, local_mbox, seria
173                 drawing->put_pixel(c);
174             }
175         }
176     }
177 }
```

Allocation site - Thread thread_video (4596) (find_and_fix_memory_errors.exe!operator()) - find_and_fix_memory_errors.cpp:170

find_and_fix_memory_errors.cpp Disassembly (find_and_fix_memory_errors.exe!0x4613) Call Stack

```
170     drawing_area * drawing = new drawing_area(startx, totaly
171     for (int x = startx ; x < stopx; x++) {
172         color_t c = render_one_pixel (x, y, local_mbox, seria
173         drawing->put_pixel(c);
174     }
175     free(drawing); //Memory Error: use delete instead of free
176     //delete drawing;
```

Call Stack

Easy Problem Management

Quickly see new problems and regressions

State	Description
New	Detected by this run
Not Fixed	Previously seen error detected by this run
Not a Problem	Set by user (tool will <u>not</u> change)
Confirmed	Set by user (tool will <u>not</u> change)
Fixed	Set by user (tool <u>will</u> change)
Regression	Error detected with previous state of "Fixed"

The screenshot shows the Intel Inspector 2017 interface. The 'Problems' table lists four memory-related issues. A yellow arrow points from the 'Regression' row in the table above to the 'New' state of problem P3 in the table below. A context menu is open over the 'New' state of P3, with the 'Change State' option selected, and a sub-menu showing 'Confirmed' as the chosen option.

ID	Type	Sources	State	Modules
P1	Mismatched allocation/deallocation	find_and_fix_memory...	Confirmed	find_and...
P2	Memory leak	find_and_fix_memory_error...	Deferred	find_and...
P3	Invalid memory access	find_and_fix_memory_errors...	New	find_and...
P4	Memory not deallocated	api.cpp; mlock.c; util.cpp; vi...	New	find_and...

- View Source
- Edit Source
- Copy to Clipboard
- Explain Problem
- Create Problem Report..
- Debug This Problem
- Change State
 - Not fixed
 - Confirmed
 - Fixed
 - Not a problem
 - Deferred
- Merge States...

[Optimization Notice](#)

Filtering - Focus on What's Important

Example: See only the errors in one source file

Before – All Errors

ID	Type	Sources	State
P1	Mismatched alloc...	find_an ...	New
P2	Mismatched alloc...	api.cpp	New
P3	Memory leak	api.cpp	Confirmed
P4	Mismatched alloc...	video.c ...	Not fixed
P5	Mismatched alloc...	video.c ...	Not fixed
P6	Mismatched alloc...	video.c ...	Not fixed

Severity	Count
Error	55 item(s)
Warning	1 item(s)

Type	Count
Invalid memory access	41 item(s)
Memory leak	1 item(s)
Memory not deallocated	11 item(s)
Mismatched allocation/dealloc...	2 item(s)

Source	Count
api.cpp	21 item(s)
find_and_fix_memory_errors.cpp	3 item(s)
util.cpp	10 item(s)
video.cpp	21 item(s)

(1) Filter – Show only one source file

After – Only errors from one source file

ID	Type	Sources	State
P1	Mismatche...	find_an ...	New
P2	Memory leak	find_an ...	Confirmed
P3	Invalid me...	find_an ...	Deferred

Severity	Count
Error	3 item(s)

Type	Count
Invalid memory access	1 item(s)
Memory leak	1 item(s)
Mismatched allocation/dealloc...	1 item(s)

Source	Count
find_and_fix_memory_errors.cpp	3 item(s)

State	Count
Confirmed	1 item(s)
Deferred	1 item(s)
New	1 item(s)

(2) Error count drops

Tip: Set the “Investigated” filter to “Not investigated” while investigating problems. This removes from view the problems you are done with, leaving only the ones left to investigate.

Incrementally Diagnose Memory Growth

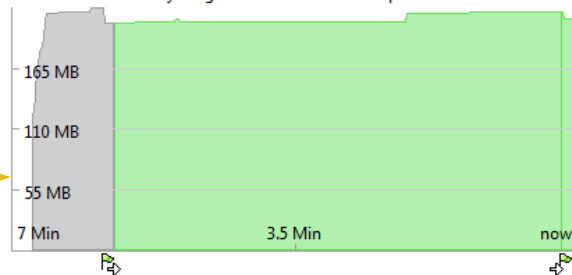
Intel® Inspector

As your app is running...

Memory usage graph plots memory growth

Memory Used by Analysis Tool and Target Application

Last recorded memory usage before collection completed: 211 MB



Select a cause of memory growth

The screenshot shows the 'Problems' window with a table of memory growth issues. The selected issue is 'Memory growth' in 'find_and_fix_memory_errors.cpp:163'.

ID	Type	Sources	Modules	Object Size	State
	Memory growth	gdiplus.dll:0x47240	gdiplus.dll	40960	New
	Memory growth	find_and_fix_memory_errors.cpp:163	find_and_fix_memory_errors.exe	90108	Not fixed
	Memory growth	find_and_fix_memory_errors.cpp:163	find_and_fix_memory_errors.exe	1802160	Not fixed
	Memory growth	find_and_fix_memory_errors.cpp:163	find_and_fix_memory_errors.exe	30036	Not fixed
	Memory growth	find_and_fix_memory_errors.cpp:163	find_and_fix_memory_errors.exe	1621944	Not fixed
	Memory growth	find_and_fix_memory_error.cpp:170	find_and_fix_memory_error.exe	40	Not fixed

The selected issue is expanded to show the code snippet and call stack:

Description	Source	Function	Module	Object Size	Offset
Allocation site	find_and_fix_memory_errors.cpp:163	operator()	find_and_fix_memory_errors.exe	90108	
161	unsigned int serial=1;				
162	unsigned int mboxsize = sizeof(unsigned int)*(max_objectid() +				
163	unsigned int * local_mbox = (unsigned int *) malloc(mboxsize);				
164					
165	for (unsigned int i=0;i<=(mboxsize/(sizeof(unsigned int)));i++				

See the code snippet & call stack

Speed diagnosis of difficult to find heap errors

Automate Regression Analysis

Command Line Interface

inspxe-cl is the command line:

- **Windows:** C:\Program Files\Intel\Inspector \bin[32|64]\inspxe-cl.exe
- **Linux:** /opt/intel/inspector_xe/bin[32|64]/inspxe-cl

Help:

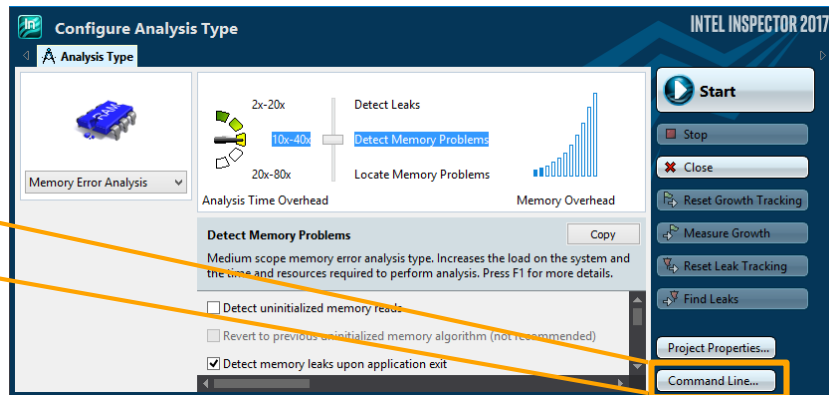
```
inspxe-cl -help
```

Set up command line with GUI

Command Line...

Command examples:

1. `inspxe-cl -collect-list`
2. `inspxe-cl -collect ti2 -- MyApp.exe`
3. `inspxe-cl -report problems`



Send results file to developer to analyze with the UI

Break At Just The Right Time

Intel® Inspector - Memory & Thread Debugger

Memory Errors

Problems		
ID	Type	Sources
P1	Mismatched allocation/deallocation	
P2	Memory leak	
P3	Invalid memory access	
P4	Memory growth	
P5	Memory growth	
P6	Memory growth	

Context menu for P3:

- View Source
- Edit Source
- Copy to Clipboard
- Explain Problem
- Create Problem Report...
- Debug This Problem

Threading Errors

Problems			
ID	Type	Sources	Modules
P1	Data race	winvideo.h	
	Data race	winvideo.h:270	
	Data race	winvideo.h:270	
	Data race	winvideo.h:201	
	Data race	winvideo.h:202	
	Data race	winvideo.h:202	

Context menu for P1:

- View Source
- Edit Source
- Copy to Clipboard
- Explain Problem
- Create Problem Report...
- Debug This Problem

Break into the debugger just before the error occurs.

Examine the variables and threads.

Diagnose the problem.

Save time. Find and diagnose errors with less effort.

Work Smarter & Faster

Intel® Inspector - Memory & Thread Debugger

Precise Error

Suppression

```
Suppression = {  
  Name = "Example",  
  Type = { uninitialized_memory_access }  
  Stacks = {  
    {  
      mod=a.out, func=update_x;  
      func=main;  
    }  
  }  
}
```

Precise, easy to edit, team shareable.
Choose which stack frame to suppress.
Eliminate the false, not the real errors.

Pause/Resume

Collection

```
__itt_suppress_push(__itt_suppress_threading_errors);  
/* Any threading errors here are ignored */  
__itt_suppress_pop();  
/* Any threading errors here are seen */
```

Speed-up analysis by limiting its scope.
Analyze only during the execution of the suspected problem.

Find and diagnose errors with less effort.

Productive Memory & Threading Debugger

Intel® Inspector

	Memory Analysis	Threading Analysis
View Context of Problem		
Stack	✓	✓
Multiple Contributing Source Locations	✓	✓
Collapse multiple “sightings” to one error (e.g., memory allocated in a loop, then leaked is 1 error)	✓	✓
Suppression, Filtering, and Workflow Management	✓	✓
Visual Studio* Integration (Windows*)	✓	✓
Command line for automated tests	✓	✓
Time Line visualization	✓	✓
Memory Growth during a transaction	✓	
Trigger Debugger Breakpoint	✓	✓

Easier & Faster Debugging of Memory & Threading Errors

[Optimization Notice](#)

Copyright © 2020, Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.

Intel® Inspector Correctness Checking for Hybrid MPI/Threading Applications

Intel® Inspector provides memory and correctness checking for threaded applications

For hybrid MPI/threading applications, the command `inspxe-cl` can be used as the MPI executable for `mpirun`:

```
mpirun -n <N> inspxe-cl -result-dir <dir> -collect <mode> -- <executable>
```

All MPI ranks will run under the control of the Intel Inspector!

Using the Intel MPI flag `-gtool` to restrict the tool to a selection of MPI ranks:

```
mpirun -gtool "inspxe-cl -collect <mode> \  
-result-dir <dir> :<rank_set>" -n <N> <executable>
```

Intel® Inspector – Collection Modes

Query Intel® Inspector collection modes

```
inspxe-cl -help collect
```

or see settings in the Intel® Inspector GUI. Available collection modes:

- mi1 Detect Leaks
- mi2 Detect Memory Problems
- mi3 Locate Memory Problems
- ti1 Detect Deadlocks
- ti2 Detect Deadlocks and Data Races
- ti3 Locate Deadlocks and Data Races

mi3 and ti3 are the most demanding memory and threading modes

Intel® Inspector -- Results

After running the MPI program, result directories will appear with the previously defined base name and indexed with the MPI rank.

Results may be viewed as ASCII output:

```
inspxe-cl -report problems -result-dir <dir>.<rank>
```

or by using the Intel® Inspector GUI:

```
inspxe-gui <dir>.<rank>
```

Results may also be transferred to a Windows* computer and viewed by the Windows* version of Intel® Inspector (using inspxe-cl with export flags to include the sources)

Additional Material

Intel® Inspector – Memory & Thread Debugger:

- [Product page](#) – overview, features, FAQs, support...
- [Training materials](#) – movies, tech briefs, documentation...

Additional Analysis Tools:

- [Intel® VTune Amplifier](#) – performance profiler
- [Intel® Advisor](#) – vector optimization and thread prototyping tool for architects

Additional Development Products:

- [Intel® Software Development Products](#)

Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation. Learn more at [intel.com](https://www.intel.com) or from the OEM or retailer.

Your costs and results may vary.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804. <https://software.intel.com/en-us/articles/optimization-notice>

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors.

Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. See backup for configuration details. For more complete information about performance and benchmark results, visit www.intel.com/benchmarks.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See configuration disclosure for details. No product or component can be absolutely secure.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

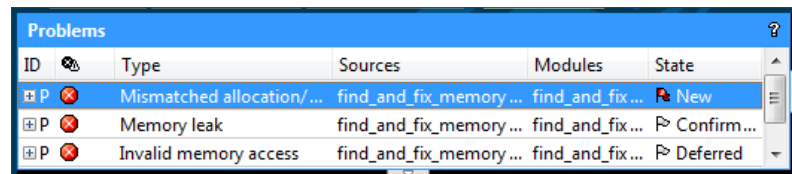
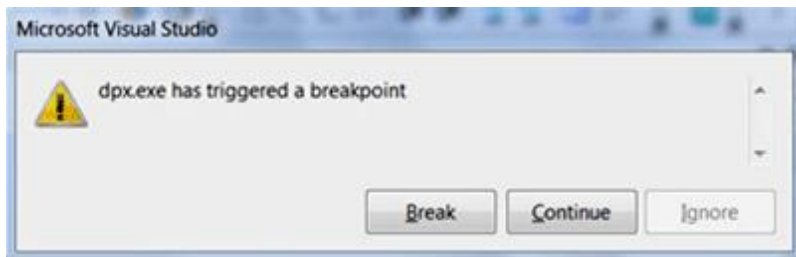
Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

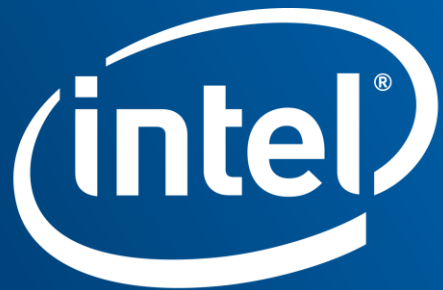
BACKUP

Pointer Checker vs. Memory Checker

Pointer Checker	Memory Checker
Recompile with Intel® Compiler	Use any build, any compiler
Lower overhead	Higher overhead
Only finds pointer errors	Finds multiple error types
One error at a time	GUI sorts multiple errors
Traceback: Source file + Line #	Traceback: Shows source code
Triggers debugger breakpoint	Triggers debugger breakpoint



Two great ways to create more reliable software



Software