

Intel Inspector XE

Memória- és szálproblémák ellenőrzése

Áfra Attila Tamás

Budapesti Műszaki és Gazdaságtudományi Egyetem
IIT / Számítógépes Grafika Csoport

Babeş-Bolyai Tudományegyetem

Bevezető

- Memória- és szálproblémák detektálására kifejlesztett **dinamikus** elemző
- Támogatott nyelvek: C, C++, C#, Fortran
- Támogatott oprendszerek: Windows, Linux
- Nagyon könnyen használható grafikus felület!
- Beépül a Visual Studio-ba
- Linuxra ingyenes*
- Legújabb: *Inspector XE 2011*, *Thread Checker* utódja

Elemzés (1)

- Az elemzendő programot nem kell újrakompilálni
- Lehet Debug és Release módban kompilált binárisokat is elemezni, de javasolt a Debug mód
- Futtatás előtt meg kell adni az elemzés típusát
- 2 fő kategória: memóriaproblémák, szálproblémák
- Alkategóriák: elemzés komplexitása
- Létre lehet hozni testreszabott elemzéstípusokat

Elemzés (2)

- Az elemzés jelentősen lelassítja a program futását:
2x-320x
- Az elemzés eredménye: probléma típusa és helye a forráskódban
- Egy problémához több kódrészlet is tartozhat (code location)
 - pl. inicializálatlan memóriához való hozzáférés:
lefoglalás helye, olvasás helye
- Rendelkezésre áll a hívási verem és az idővonal is

Memóriaproblémák

- Szivárgás (memory leak)
- Rosszul párosított lefoglalás/felszabadítás
 - pl. *new[]* és *delete*
- Hibás címről való felszabadítás
 - pl. többszörös felszabadítás
- Inicializálatlan memóriához való hozzáférés
- Hibás címhez való hozzáférés

Szálproblémák

- Holtpont (deadlock)
- Lock hierarchy violation
 - A szálak eltérő sorrendben akarnak birtokolni egymásba ágyazott zárat
- Adatverseny (data race)
 - write → write, read → write, write → read
- Potential privacy infringement
 - Egy szál hozzáfér egy másik szál verméhez

Az elemző használata

New Inspector XE Result × test.cpp

Intel Inspector XE 2011

Configure Analysis Type

Analysis Type

- Memory Error Analysis
 - Detect Leaks
 - Detect Memory Problems
 - Locate Memory Problems**
- Threading Error Analysis
 - Detect Deadlocks
 - Detect Deadlocks and Data Races
 - Locate Deadlocks and Data Races
- Custom Analysis Types

Locate Memory Problems

Widest scope memory error analysis type. Maximizes the load on the system. Maximizes the time required to perform the analysis. Maximizes the chances the analysis will fail because the system may run out of resources. Press F1 for more details.

Detect resource leaks

Stack frame depth: 16

Analyze stack accesses

Remove duplicates

Details

Detect memory leaks:	Yes
Detect resource leaks:	Yes
Detect invalid/uninitialized accesses:	Yes
Analyze stack accesses:	No
Enable enhanced dangling pointer check:	Yes
Byte limit before reallocation:	1 Mb
Enable guard zones:	Yes
Guard zone byte size:	32 bytes
Stack frame depth:	16
Remove duplicates:	Yes

Start

Stop

Set Transaction Start

Set Transaction End

Close

Project Properties

Show Command Line

Példa

```
test.cpp X
(Global Scope)
#include <iostream>
using namespace std;
int main()
{
    int* data = new int[20];

    data[5] = 144;
    data[11] = 2;

    int i;
    cin >> i;

    cout << data[i] << endl;
    cout << data[50] << endl;
    return 0;
}
```


Detektált memóriaproblémák

The screenshot displays the Intel Inspector XE 2011 interface for analyzing memory problems. The main window shows a list of detected issues, with the 'Summary' tab selected. The 'Problems' table lists three issues: P1 (Invalid memory access), P2 (Uninitialized memory access), and P3 (Memory leak). P2 and P3 are marked as 'Not fixed'. The 'Code Locations' pane shows the source code for the 'main' function in 'test.cpp', with lines 7 and 15 highlighted. Line 7 is the allocation site for the memory leak, and line 15 is the source of the uninitialized memory access. The 'Filters' pane on the right provides a summary of the detected issues, showing 3 items for each severity, problem, source, module, state, and investigation status.

Intel Inspector XE 2011

Locate Memory Problems

Target Analysis Type Collection Log Summary

ID	Problem	Sources	Modules	Object Size	State
P1	Invalid memory access	test.cpp	test.exe		New
P2	Uninitialized memory access	test.cpp	test.exe		Not fixed
P3	Memory leak	test.cpp	test.exe	80	Not fixed

Code Locations

ID	Description	Source	Function	Module	Object Size	Offset
X3	Allocation site	test.cpp:7	main	test.exe		
X2	Read	test.cpp:15	main	test.exe		36

```
5 int main()
6 {
7 int* data = new int[20];
8
9 data[5] = 144;
13 cin >> i;
14
15 cout << data[i] << endl;
16 cout << data[50] << endl;
17 return 0;
```

Filters

Severity	Count
Error	3 item(s)

Problem

Problem	Count
Memory leak	1 item(s)
Invalid memory access	1 item(s)
Uninitialized memory access	1 item(s)

Source

Source	Count
test.cpp	3 item(s)

Module

Module	Count
test.exe	3 item(s)

State

State	Count
New	1 item(s)
Not fixed	2 item(s)

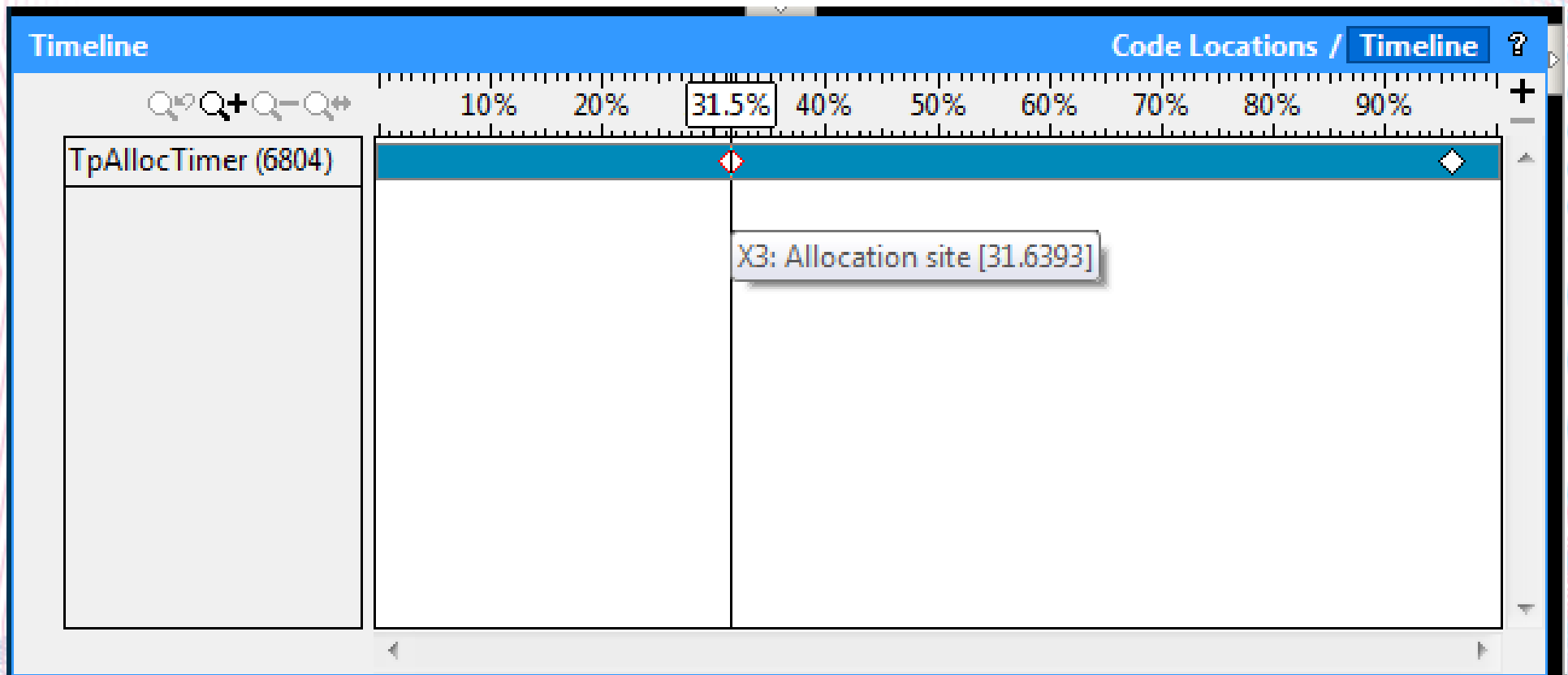
Suppressed

Suppressed	Count
Not suppressed	3 item(s)

Investigated

Investigated	Count
Not investigated	3 item(s)

Idővonal



Hibaforrások

Intel Inspector XE 2011

Locate Memory Problems

Target Analysis Type Collection Log Summary Sources

Focus Code Location: test.cpp:15 - Read

```
12 int i;
13 cin >> i;
14
15 cout << data[i] << endl;
16 cout << data[50] << endl;
17 return 0;
18 }
```

Call Stack

- test.exe!main - test.cpp:15
- test.exe!_tmainCRTStartup - crtexe.c:555
- test.exe!mainCRTStartup - crtexe.c:370
- kernel32.dll!BaseThreadInitThunk - kernel32.dll:91435
- ntdll.dll!RtlUserThreadStart - ntdll.dll:181535

Related Code Location: test.cpp:7 - Allocation site

```
4
5 int main()
6 {
7 int* data = new int[20];
8
9 data[5] = 144;
10 data[11] = 2;
11
```

Call Stack

- test.exe!main - test.cpp:7
- test.exe!_tmainCRTStartup - crtexe.c:555
- test.exe!mainCRTStartup - crtexe.c:370
- kernel32.dll!BaseThreadInitThunk - kernel32.dll:91435
- ntdll.dll!RtlUserThreadStart - ntdll.dll:181535

Code Locations

ID	Description	Source	Function	Module	Object Size	Offset
X3	Allocation site	test.cpp:7	main	test.exe		
X2	Read	test.cpp:15	main	test.exe		36

Összefoglalás

- Az Intel Inspector XE egy egyszerű de nagyon hasznos hibaelemző
- Gyorsan lokalizálható számos súlyos, gyakran előforduló memória- és szálprobléma
- Olyan hibákat is észrevehet, amire egy statikus elemző nem képes
- Viszont ez fordítva is igaz!

Köszönöm a figyelmet!