Profile

Típusok I.

- Sampling
  - Bele-belenéz a processbe, hogy éppen hol tart.
  - Gyors, de kevésbé pontos
  - Mi van, ha vár valamilyen szinkronizációs objektumra?
- Instrumenting
  - Beleír a kódba
  - Pontos, de lassú
  - Eljárásokat / programsorokat is lehet vele mérni
  - Figyelembe veszi a várakozás időket is

Típusok II.

```
int hivo1() {hivott();}
int hivo2() {hivott();}
```

- Call graph
  hivo1, hivo2, hivott
- Call tree
  hivo1->hivott, hivo2->hivott

Eszközök

OSX, Linux: gprof

- **-pg**-vel kell fordítani és linkelni
- futtatni
- **gmon.out** lesz a kimenet
- **gprof**-fal megnézni
- nagyon bő kimenet

Windows: Visual studio

- premiumban és ultimate-ben
- van sok más olcsóbb tool is

Debug

Miért?

- Crash
  - Leggyakrabban memória miatt
- Fagyás
  - Leggyakrabban deadlock miatt
- Nem jól működik

Mivel?

Windows

- Visual studio
- WinDbg
  - Nagyon dicsérik

Linux, OSX

- GDB és frontendjei
  - XCode (OSX)
  - Qt Creator
  - Eclipse
  - ddd

Általános feature-ök

- Memória
- Regiszterek
- Szálak
- Call stackek
- Változók értelmezése
- Watch
- Break/Trace point(+ feltételes)

Spéci featurök

- Adatok módosítása
- Kód módosítása
- Át lehet tenni a jelenlegi utasítást
- Külön debug output(MS):

  `OutputDebugString`

Hogyan?

- Debuggerrel indított process
- Rácsatlakozni futó processre (attach)
  - Debugger várás: **`IsDebuggerPresent`**(MS)
- Remote debugger (MS remote debugger, vagy gdbserver)
- Crashdump

strace, ltrace

Használtat:

strace bin/valami

- strace
  - system trace
  - rendszer hívásokat lehet vele követni
  - .so-k betöltése, fájlrendszerre írás
- ltrace
  - lib hívásokat lehet követni
  - nagyon bő kimenet

strace, ltrace példa kimenet

strace példa

```
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\0\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\200\30\2\0\0\0\0\0"..., 832) =
832
fstat(3, {st_mode=S_IFREG|0755, st_size=1811160, ...}) = 0
```

ltrace példa

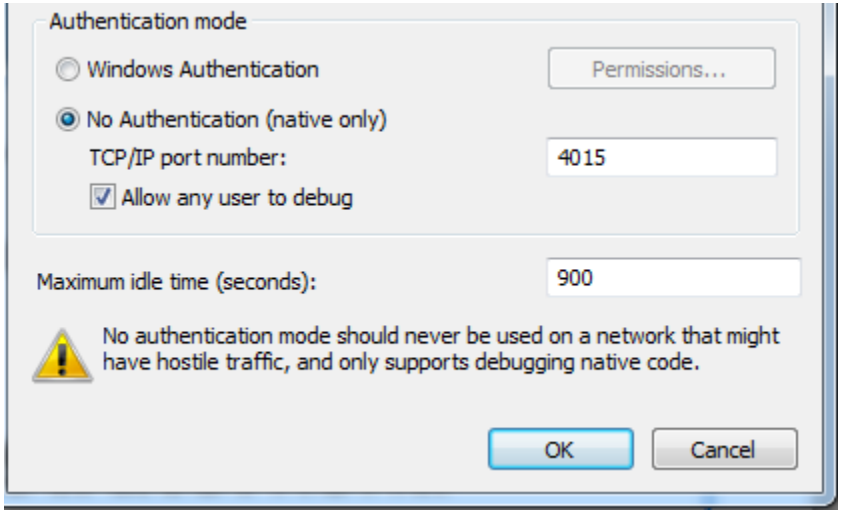
```
_Znwm(8, 132864, 0x3fffffffffffffff, 64, 3)                    = 0xbb9960
memmove(0xbb9960, 0xbb9030, 8, 2, 0xbb9960)                    = 0xbb9960
_ZSt29_Rb_tree_insert_and_rebalancebPSt18_Rb_tree_node_baseS0_RS_(1, 0xbb9910, 0x607508,
0x607508, 0xbb9968) = 0xbb9960
_Znwm(64, 0, 0xbb9910, 0x7ffff7474880, 0xbb9968)               = 0xbb9980
```

Remote debuggolás - MS

Visual studio remote debugger path:

```
\Program Files (x86)\Microsoft Visual Studio 10.0\Common7\IDE\Remote Debugger\[x64|x86]\msvsmon.exe
```

Távoli gépen elindítani **administrator**ként.



Remote debuggolás - unix

Távoli gép:
gdbserver :12345 ./testProg

Fejlesztői gép:
gdb ./testProg
target remote X.X.X.X:12345

Crashdump

- Hibás utasításkor ("elszáll a program") keletkezik
- A folyamathoz tartozó memóriakép, regiszter állapotok
- Létrehozási helyek
  - Linux: ./core.<PID> (ulimit -c unlimited)
  - Mac: /cores/core.<PID> (ulimit -c unlimited)
  - Windows:
    - %SystemRoot%\MEMORY.DMP
    - %SystemRoot%\Minidump
    - .\diag\<Exe-Date>.dmp
    - `SetUnhandledExceptionFilter (ExceptionFilterProc);`

Crashdump - elemzés

Betöltés:

- GDB: gdb ./testProg core.1234
- Visual Studio: Drag&Drop
- Qt Creator:
  Debug > Start Debugging > Load Core File ...

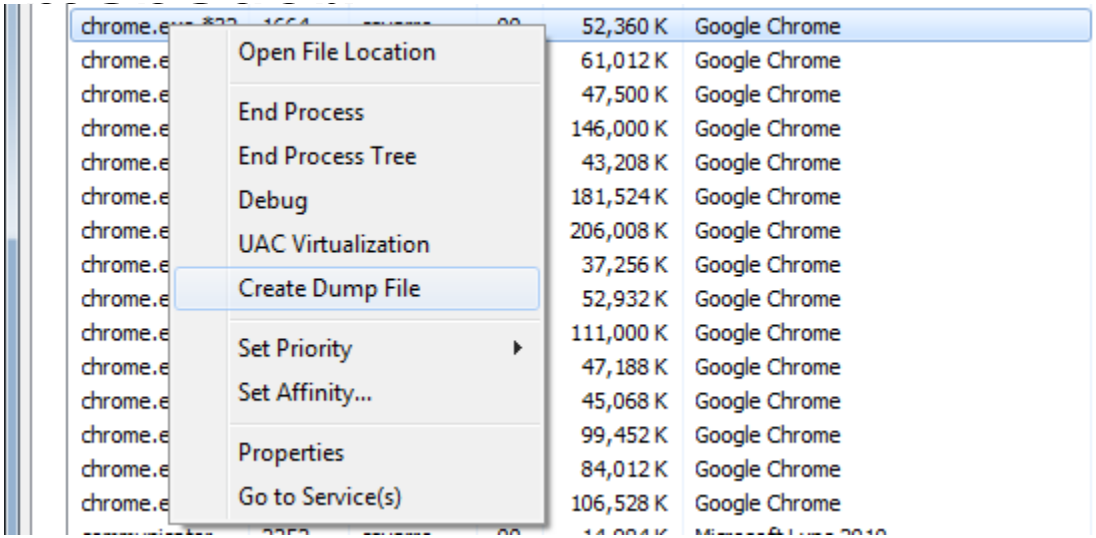
Innen szokásos debugolás.

Debug build és meglévő forrás esetén igazán eredményes.

Időzítési hibákat néha csak így lehet megfogni.

Coredump készítés - futás közben

Windows - Task



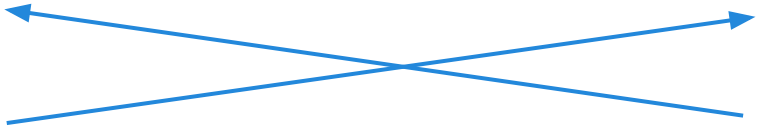
Linux:

`gcore <pid>`

(rootként)

Deadlockok

Szál1
- Mutex1
- Mutex2



Szál2
- Mutex2
- Mutex1

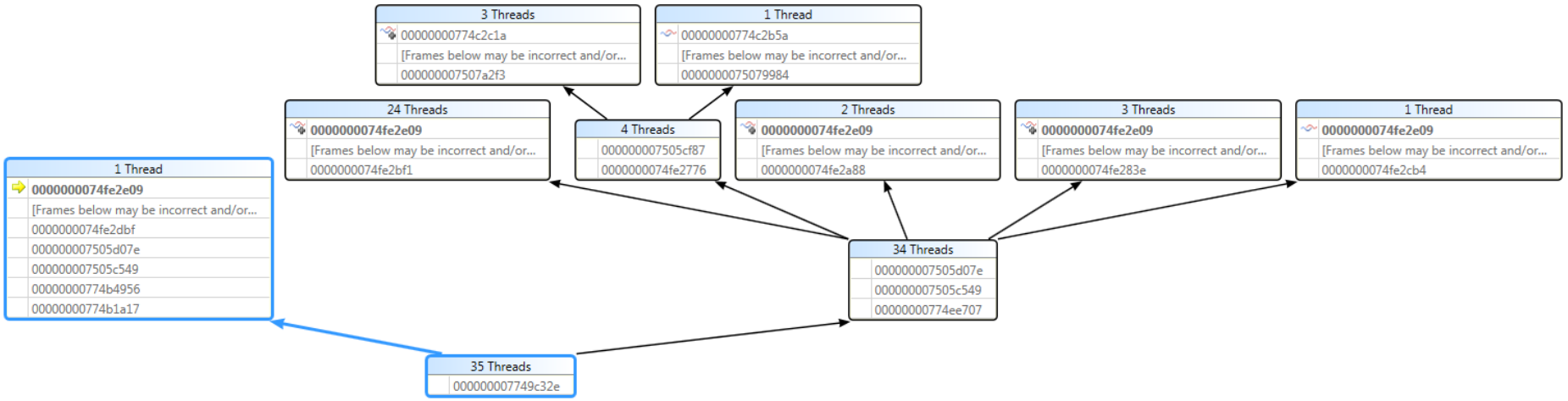
Olyan szinkronizációs objektumra várnak, amit a másik éppen fog.
Általában nehéz reprodukálni.
Viszont kényelmesen rá lehet attacholni egy debuggerrel

Deadlockok

GDB és VS is jelzi, ha deadlock van

1. Mire várnak a szálak?
2. Melyik szál fogja az adott objektumot?
3. Az a szál mire vár?
4. GOTO 2.pont

Visual studio 10+



deadlock ellenszer

- A lockokat mindig ugyabban a sorrendben kell megfogni.
- Többszálas, taskos design.
- Minimalizálni a szálak között megosztott változókat.
- Resource-okat és vele a szinkronizációt interface-ek mögé rejteni

Memory fence

Leggyakrabbak(VS):

`0xCCCCCCCC` : Inicializálatlan **stack**

`0xCDCDCDCD` : Inicializálatlan **heap**

`0xFDFDFDFD` : "No man's land" a **heap**-en

`0xDDDDDDDD` : Felszabadított **heap**

`0xFEEEFEEE` : Felszabadított **heap**

GCC: Stack guardok vannak. Hasonló.

Valgrind

- memória és teljesítmény ellenőrző programcsomag

- Hátrányok:
  - csak Linux és OS X
  - Debug módban kell fordítani a programot (lassú!)
  - az eredeti program nem fut közvetlenül a processzoron (még lassabb!)
  - néha túlságosan is bőbeszédű kimenet

Valgrind - eszközök

- memcheck
  - memória szivárgás, túlírás, túlolvasás
- cachegrind
  - betöltött és végrehajtott utasítások
- callgrind
  - cachegrind kiegészítése hívási láncokkal
- massif
  - a heap memória használata
- helgrind
  - szálak közötti versenyhelyzetek
- drd
  - többszálú C és C++ programok hibái

Valgrind - memcheck futtatás

```
./debugExample
```

helyett:

```
valgrind --tool=memcheck .
/debugExample
```

vagy, mivel a memcheck az alapértelmezett:

```
valgrind ./debugExample
```

Valgrind - memcheck

```
==3442== Memcheck, a memory error detector
==3442== Copyright (C) 2002-2012, and GNU GPL'd, by Julian Seward et al.
==3442== Using Valgrind-3.8.1 and LibVEX; rerun with -h for copyright info
==3442== Command: ./debugExample
==3442==

==3442== Invalid write of size 4
==3442==    at 0x8048ED1: SloppyVector::push_back(int) (main.cpp:27)
==3442==    by 0x8048D80: main (main.cpp:65)
==3442==  Address 0x434b02c is 0 bytes after a block of size 4 alloc'd
==3442==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3442==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3442==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3442==    by 0x8048D80: main (main.cpp:65)
```

Az első hiba

Valgrind - memcheck

```
==3442== Invalid write of size 4
==3442==    at 0x8048ED1: SloppyVector::push_back(int) (main.cpp:27)
==3442==    by 0x8048D80: main (main.cpp:65)
==3442==  Address 0x434b02c is 0 bytes after a block of size 4 alloc'd
==3442==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3442==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3442==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3442==    by 0x8048D80: main (main.cpp:65)
```

A program azonosítója (pid).

Valgrind - memcheck

```
==3442== Invalid write of size 4
==3442==    at 0x8048ED1: SloppyVector::push_back(int) (main.cpp:27)
==3442==    by 0x8048D80: main (main.cpp:65)
==3442==  Address 0x434b02c is 0 bytes after a block of size 4 alloc'd
==3442==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3442==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3442==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3442==    by 0x8048D80: main (main.cpp:65)
```

A hiba leírása.

Valgrind - memcheck

```
==3442== Invalid write of size 4
==3442==    at 0x8048ED1: SloppyVector::push_back(int) (main.cpp:27)
==3442==    by 0x8048D80: main (main.cpp:65)
==3442==  Address 0x434b02c is 0 bytes after a block of size 4 alloc'd
==3442==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3442==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3442==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3442==    by 0x8048D80: main (main.cpp:65)
```

A hiba bekövetkezésének a helye.

Valgrind - memcheck

```
==3442== Invalid write of size 4
==3442==    at 0x8048ED1: SloppyVector::push_back(int) (main.cpp:27)
==3442==    by 0x8048D80: main (main.cpp:65)
==3442==  Address 0x434b02c is 0 bytes after a block of size 4 alloc'd
==3442==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3442==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3442==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3442==    by 0x8048D80: main (main.cpp:65)
```

Az utasítás címe. (Trükkös hibákhoz)

Valgrind - memcheck

```
==3442== Invalid write of size 4
==3442==    at 0x8048ED1: SloppyVector::push_back(int) (main.cpp:27)
==3442==    by 0x8048D80: main (main.cpp:65)
==3442==  Address 0x434b02c is 0 bytes after a block of size 4 alloc'd
==3442==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3442==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3442==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3442==    by 0x8048D80: main (main.cpp:65)
```

A forrásban hol találjuk.

Valgrind - memcheck

```
==3442== Invalid write of size 4
==3442==    at 0x8048ED1: SloppyVector::push_back(int) (main.cpp:27)
==3442==    by 0x8048D80: main (main.cpp:65)
==3442==  Address 0x434b02c is 0 bytes after a block of size 4 alloc'd
==3442==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3442==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3442==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3442==    by 0x8048D80: main (main.cpp:65)
```

A hibához vezető körülmény vagy ok.

Valgrind - memcheck

```
==3442== Invalid write of size 4
==3442==    at 0x8048ED1: SloppyVector::push_back(int) (main.cpp:27)
==3442==    by 0x8048D80: main (main.cpp:65)
==3442==  Address 0x434b02c is 0 bytes after a block of size 4 alloc'd
==3442==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3442==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3442==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3442==    by 0x8048D80: main (main.cpp:65)
```

Ezért találja meg a valgrind. Nem a hiba része.

Valgrind - memcheck

```
==3442== Invalid write of size 4
==3442==    at 0x8048ED1: SloppyVector::push_back(int) (main.cpp:27)
==3442==    by 0x8048D80: main (main.cpp:65)
==3442==  Address 0x434b02c is 0 bytes after a block of size 4 alloc'd
==3442==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3442==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3442==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3442==    by 0x8048D80: main (main.cpp:65)
```

Ez itt lefoglalt 4 bájtos blokk íródott túl.

Valgrind - memcheck

Memóriaszivárgás is van a programban.

```
==3442== HEAP SUMMARY:
==3442==     in use at exit: 20 bytes in 3 blocks
==3442==   total heap usage: 4 allocs, 1 frees, 28 bytes allocated
==3442==
==3442== LEAK SUMMARY:
==3442==    definitely lost: 20 bytes in 3 blocks
==3442==    indirectly lost: 0 bytes in 0 blocks
==3442==      possibly lost: 0 bytes in 0 blocks
==3442==    still reachable: 0 bytes in 0 blocks
==3442==         suppressed: 0 bytes in 0 blocks
==3442== Rerun with --leak-check=full to see details of leaked memory
==3442==
```

Valgrind - memcheck

Memóriaszivárgás is van a programban.

```
==3442== HEAP SUMMARY:
==3442==     in use at exit: 20 bytes in 3 blocks
==3442==   total heap usage: 4 allocs, 1 frees, 28 bytes allocated
==3442==
==3442== LEAK SUMMARY:
==3442==    definitely lost: 20 bytes in 3 blocks
==3442==    indirectly lost: 0 bytes in 0 blocks
==3442==      possibly lost: 0 bytes in 0 blocks
==3442==    still reachable: 0 bytes in 0 blocks
==3442==         suppressed: 0 bytes in 0 blocks
==3442== Rerun with --leak-check=full to see details of leaked memory
==3442==
```

Valgrind - memcheck

```
==3464== 8 bytes in 1 blocks are definitely lost in loss record 2 of 3
==3464==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3464==    by 0x8048F18: SloppyVector::increment() (main.cpp:40)
==3464==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3464==    by 0x8048D94: main (main.cpp:66)
```

Egy 8 bájtos memóriaterületre mutató mutató felülírodott, ezért ha akarnánk se tudnánk felszabadítani.

Valgrind - memcheck

```
==3464== 8 bytes in 1 blocks are definitely lost in loss record 2 of 3
==3464==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3464==    by 0x8048F18: SloppyVector::increment() (main.cpp:40)
==3464==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3464==    by 0x8048D94: main (main.cpp:66)
```

Megint csak a valgrindes megvalósítás miatt van itt ez a sor. Egy tömböt nem szabadítottunk fel.

Valgrind - memcheck

```
==3464== 8 bytes in 1 blocks are definitely lost in loss record 2 of 3
==3464==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3464==    by 0x8048F18: SloppyVector::increment() (main.cpp:40)
==3464==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3464==    by 0x8048D94: main (main.cpp:66)
```

Itt hoztuk létre a tömböt.

Valgrind - memcheck

```
==3480== Use of uninitialised value of size 4
==3480==    ...
==3480==    by 0x8048DD0: main (main.cpp:70)
==3480==  Uninitialised value was created by a heap allocation
==3480==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3480==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3480==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3480==    by 0x8048D80: main (main.cpp:65)
```

Úgy használtunk egy 4 bájtos memóriaterületet, hogy nem adtunk értéket neki előtte.

Valgrind - memcheck

```
==3480== Use of uninitialised value of size 4
==3480==    ...
==3480==    by 0x8048DD0: main (main.cpp:70)
==3480==  Uninitialised value was created by a heap allocation
==3480==    at 0x402B774: operator new[](unsigned int) (in
/usr/lib/valgrind/vgpreload_memcheck-x86-linux.so)
==3480==    by 0x8048EF0: SloppyVector::increment() (main.cpp:35)
==3480==    by 0x8048EC0: SloppyVector::push_back(int) (main.cpp:25)
==3480==    by 0x8048D80: main (main.cpp:65)
```

Itt hoztunk létre.

Statikus elemző

Statikus elemző

Nem a program futtatásával, hanem a forráskód elemzésével világítanak rá hibákra.

A projekt elemzése a fordításnál olykor lényegesen hosszabb ideig tart.

Nem tökéletes. Csak a belekódolt hibákat találja meg, de még az is előfordulhat, hogy téved és hibát jelez.

scan-build / scan-view

A clang fordítócsomag része
Az XCode beépítve tartalmazza (Analyze)

Használat (cmake):

- `scan-build cmake ../staticExample`
- `scan-build make`
- `scan-view /tmp/scan-build-*`

scan-view

Bug Summary

Bug Type	Quantity	Display?
All Bugs	**3**	☑
Dead store		
Dead assignment	1	☑
Dead initialization	1	☑
Logic error		
Dereference of null pointer	1	☑

Reports

Bug Group	Bug Type ▾	File	Line	Path Length			
Dead store	Dead assignment	main.cpp	12	1	View Report	Report Bug	Open File
Dead store	Dead initialization	main.cpp	7	1	View Report	Report Bug	Open File
Logic error	Dereference of null pointer	main.cpp	16	3	View Report	Report Bug	Open File

scan-view

Annotated Source Code

```
1  #include <stdio.h>
2
3  int main()
4  {
5      int* nullPointer = NULL;
```

1 Variable 'nullPointer' initialized to a null pointer value →

```
6      int evenNumber=2;
7      int neverUsed=evenNumber;
8      bool wideScopeVariable;
9
10     if (evenNumber%2 == 1)
```

2 ← Taking false branch →

```
11     {
12         wideScopeVariable = true;
13         nullPointer = new int[evenNumber];
14     }
15
16     nullPointer[evenNumber-1]=evenNumber;
```

3 ← Array access (from variable 'nullPointer') results in a null pointer dereference

```
17
18     delete nullPointer;
19
20     return 0;
21 }
```

cppcheck

Nem célja a fordító helyettesítése.

Azokról a hibákról jelent, ami felett a fordító átsiklik.

Leginkább az IDE-vel összekapcsolva használható ki.

Windowson külön GUI.

cppcheck

Feature-ök:

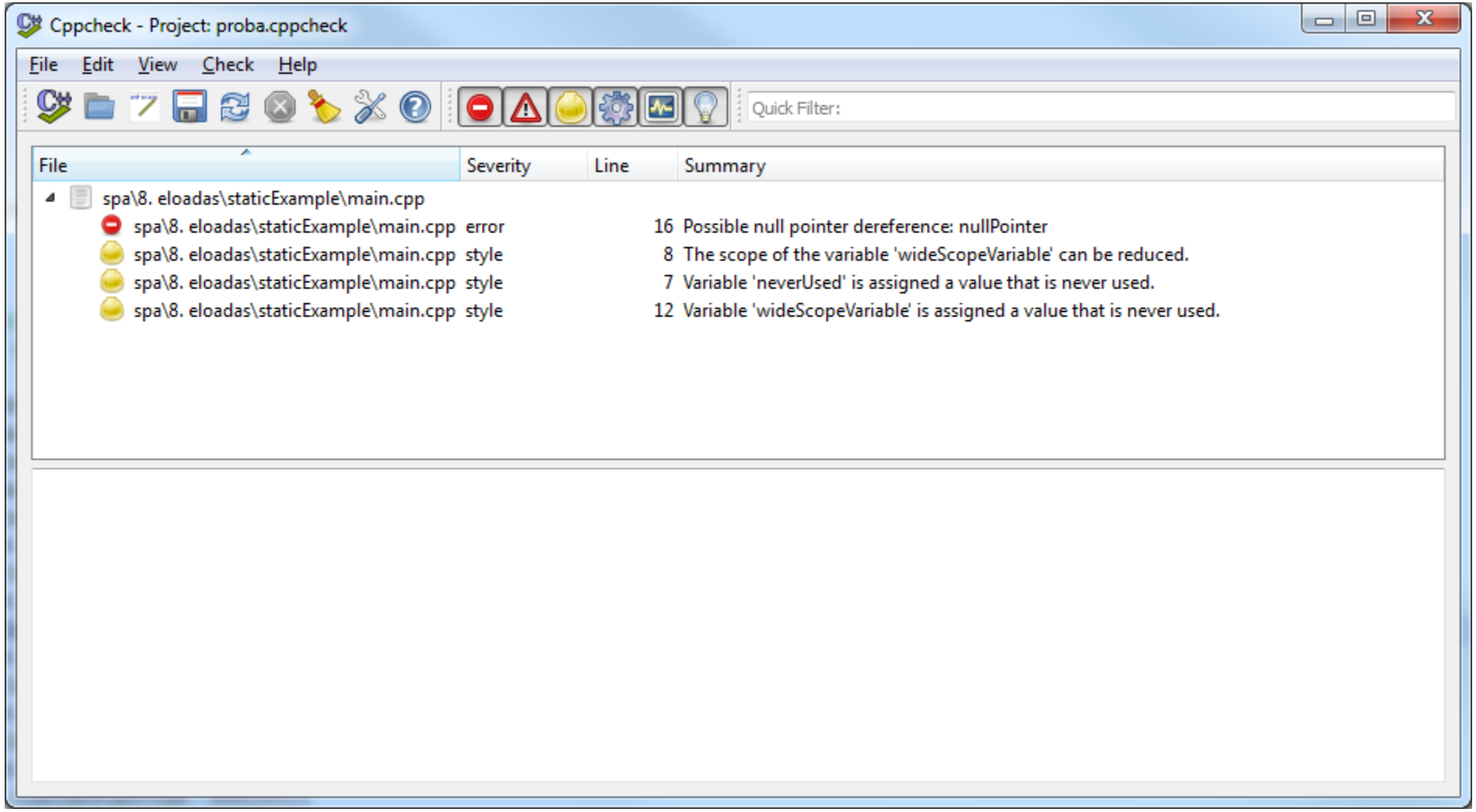
- Bounds checking
- Exception safety
- Memory leak check
- Elavult függvények
- STL helyes használat
- Inicializálatlan változók
- Nem használt függvények

cppcheck

```
cppcheck --enable=all --inconclusive --
std=posix main.cpp
```

```
Checking main.cpp...
[main.cpp:8]: (style) The scope of the variable
'wideScopeVariable' can be reduced.
[main.cpp:7]: (style) Variable 'neverUsed' is assigned a
value that is never used.
[main.cpp:12]: (style) Variable 'wideScopeVariable' is
assigned a value that is never used.
[main.cpp:16]: (error) Possible null pointer dereference:
nullPointer
Checking usage of global functions..
```

cppcheck


Cppcheck - Project: proba.cppcheck
File Edit View Check Help
Quick Filter:
File
Severity
Line
Summary
spa\8. eloadas\staticExample\main.cpp
spa\8. eloadas\staticExample\main.cpp error 16 Possible null pointer dereference: nullPointer
spa\8. eloadas\staticExample\main.cpp style 8 The scope of the variable 'wideScopeVariable' can be reduced.
spa\8. eloadas\staticExample\main.cpp style 7 Variable 'neverUsed' is assigned a value that is never used.
spa\8. eloadas\staticExample\main.cpp style 12 Variable 'wideScopeVariable' is assigned a value that is never used.

cpplint

A Google fejleszti

Egy python script

- Stílus ellenőrzés
- const, explicit használat ellenőrzés
- 0 NULL vagy \0 helyett
- Nem triviális függvények a headerben

cpplint példa

- No copyright message found. You should have a line: "Copyright [year] <Copyright Owner>" [legal/copyright] [5]
- Streams are highly discouraged. [readability/streams] [3]
- Lines should be <= 80 characters long [whitespace/line_length] [2]
- Add #include <vector> for vector<> [build/include_what_you_use] [4]
- Do not leave a blank line after "private:" [whitespace/blank_line] [3]
- { should almost always be at the end of the previous line [whitespace/braces] [4]

Források

- http://stackoverflow.com/questions/127386/in-visual-studio-c-what-are-the-memory-allocation-representations
- http://en.wikipedia.org/wiki/Magic_number_(programming)
- http://valgrind.org/
- http://support.microsoft.com/kb/315263
- http://clang-analyzer.llvm.org/
- http://cppcheck.sourceforge.net/
- http://google-styleguide.googlecode.com/svn/trunk/cpplint/cpplint.py