



## Not about what we already know...

- Be more defensive
  - /analyze (PREfast)
  - Lint
  - /RTC compiler flags VS runtime checks
    - Uninitialized variable
    - Smaller type checks (conversion to)
    - Stack frames
    - Security checks
  - ASSERT / VERIFY macros
  - Pre/Post condition checks
  - Debug builds, `_SECURE_SCL`
  - /GS (security buffer checking)
  - Unit tests

© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

©2003 M. Plonsky



2

## Golden rules

- You are guilty until proven innocent
- Always generate debugging symbols
- The symbol server is your best friend
- All access violations are deadly
- Save a .DMP file
- Have sharp tools [www.sysinternals.com](http://www.sysinternals.com) and “Debugging Tools for Windows”, VS2005



© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

3

## Not so obvious

- Unoptimized can be better than DEBUG
- Remote debugging can be better than local
- Don't use the debugger that comes with the compiler -- use the latest debugger
- Global variables can be good

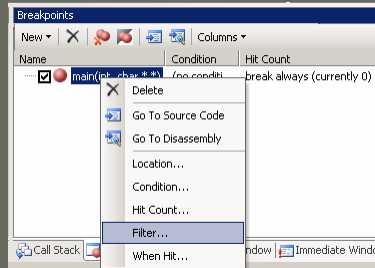


© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

4

## Use the latest debugger

- Use the latest debugger, but keep your build environment.
- What extra goodies are available?



VS 2005 goodies:

- Filters for breakpoints (tid, pid, machine process or thread name)  
See SetThreadName in MSDN.
- Run script or print values when hit (and then stop or continue)
- Proper symbol server support
- Source server support

© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

5

## Debugging nightmares

A warning to architects; how to multiply your bugs.

- All we have to do is integrate our C++ widgets, with our JScript script applets, our Java beans, and our XSL transforms, and DHTML front end, and our .NET assemblies...
- Well in theory....
- But what we get is a dangerous mess
- Be careful how many technologies you use

© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

6



## How do I ...

1. Debug what corrupted my 'this' pointer
2. Debug service startup code
3. Find what's eating all that CPU time
4. Debug a deadlock
5. Find what corrupted my heap
6. Find the cause of leaks
7. Gather .DMP files



© Mark Bartosik, www.bugbrowser.com

©2005 M. Plonsky

## How do I debug an invalid *this* ?

- What zapped my 'this pointer'?
- What corrupted my object?
- Or is my debugger a liar?

```
class my_class_t
{
    void foo(int first, int second);
};

caller                                     callee
push second                               push ebp
push first                                mov ebp, esp
mov ecx, this                              mov esi, ecx
call my_class_t::foo                       xor ebx, ebx
mov ecx, ebx                               ← ZAP
```



© Mark Bartosik, www.bugbrowser.com

8

### Default Release build with /O2 (max speed)

```

namespace globals
{
    my_class_t * ptr = 0;
}

struct my_class_t
{
    virtual ~my_class_t(){};
    void foo(int first, int second);

    std::vector<int> m_v;
};

void my_class_t::foo(int first, int second)
{
    m_v.push_back(first);
    m_v.push_back(second);
    // With a break point (or crash) here, look at "my_obj" and "globals::ptr"
}

_declspec(noinline) void some_intermediate_function( my_class_t & obj)
{
    globals::ptr = &obj;
    obj.foo(1,2);
}

int main(int argc, char * argv[])
{
    my_class_t my_obj;
    some_intermediate_function(my_obj);
}

```

Name	Value
&my_obj	0x0012ff5c {m_v=[870687]}(-2062236541,-2062236541)
__vfptr	0x00000001
m_v	[870687](-2062236541,1087939520)
globals::ptr	0x0012ff60 {m_v=[2](1,2) }
__vfptr	0x00403188 const my_class_t::
m_v	[2](1,2)

Name	Value	Type
&my_obj	0x0012ff5c {m_v=[870687]}(-2062236541,-2062236541); my_class_t	my_class_t
__vfptr	0x00000001	*
m_v	[870687](-2062236541,1087939520,14240) std::vector<int>	std::vector<int>
globals::ptr	0x0012ff60 {m_v=[2](1,2) }	my_class_t
__vfptr	0x00403188 const my_class_t:: 'vftable'	*
m_v	[2](1,2)	std::vector<int>

© Mark Bartosik, www.bugbrowser.com 9

### Default Release built with /Od (optimization disabled)

```

namespace globals
{
    my_class_t * ptr = 0;
}

struct my_class_t
{
    virtual ~my_class_t(){};
    void foo(int first, int second);

    std::vector<int> m_v;
};

void my_class_t::foo(int first, int second)
{
    m_v.push_back(first);
    m_v.push_back(second);
    // With a break point (or crash) here, look at "my_obj" and "globals::ptr"
}

_declspec(noinline) void some_intermediate_function( my_class_t & obj)
{
    globals::ptr = &obj;
    obj.foo(1,2);
}

int main(int argc, char * argv[])
{
    my_class_t my_obj;
    some_intermediate_function(my_obj);
}

```

Name	Value
&my_obj	0x0012ff5c {m_v=[2](1,2) }
__vfptr	0x00403188 const my_class_t::
m_v	[2](1,2)
globals::ptr	0x0012ff5c {m_v=[2](1,2) }
__vfptr	0x00403188 const my_class_t::
m_v	[2](1,2)

Name	Value	Type
&my_obj	0x0012ff5c {m_v=[2](1,2) }	my_class_t
__vfptr	0x00403188 const my_class_t:: 'vftable'	*
m_v	[2](1,2)	std::vector<int>
globals::ptr	0x0012ff5c {m_v=[2](1,2) }	my_class_t
__vfptr	0x00403188 const my_class_t:: 'vftable'	*
m_v	[2](1,2)	std::vector<int>

© Mark Bartosik, www.bugbrowser.com 10

## How do I debug a service?

- Debug startup code in a service?

```
::MessageBox(NULL, "Service", "Debug Me",  
            MB_SERVICE_NOTIFICATION | MB_OK );
```

You've got 30 seconds!

Before the service control manager terminates a non responsive process

Magic registry entry: ServicesPipeTimeout

```
symchk /ip 612 -s SRV*c:\websymbols*http://msdl.microsoft.com/download/symbols
```

```
cdb -pv -pn services.exe -c ".dump /ma c:\services.dmp; .detach; q"
```

```
cdb -z c:\services.dmp -c "x services!*timeout*; q"
```

```
cdb -pv -pn services.exe -y SRV*c:\websymbols  
-c ".symopt+ 0x400; ed services!g_dwScPipeTransactTimeout 0n60000 ; .detach; q"
```

```
NT_SYMBOL_PATH=SRV*c:\websymbols*http://msdl.microsoft.com/download/symbols
```

Alternatively call:

SetServiceStatus with SERVICE\_START\_PENDING and a hint of 60000 and incremented checkpoint value.

## Non invasive debugging

**TIP:** -pv == Non invasive debugging

- Only one debugger can be attached at a time, right?
- If the debugger dies the process dies, right?
- WinDbg and CDB support limited debugging without attaching as a debugger.
- Can use WinDbg in conjunction with Visual Studio
- Can probe critical applications safely
- If the application is completely frozen and the debugger cannot launch a break thread necessary for a true attach. In this case typically the *loader lock* is held.
- Does not affect timings and hide bugs.

## But, it works under the Debugger!! #\$\$%☹

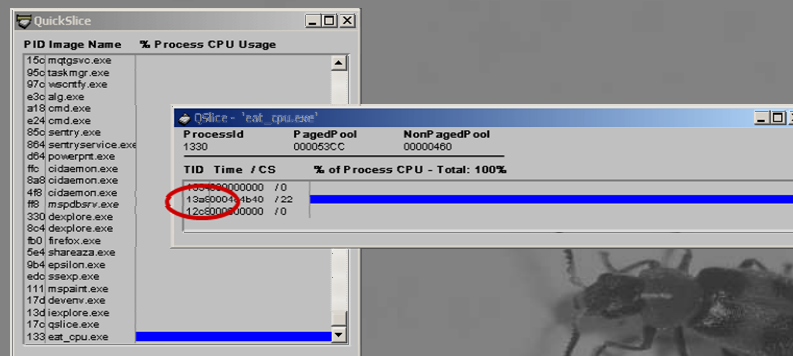
- Problems
  - Program timing is affected:
    - Thread start / stop
    - Module load / unload
    - OutputDebugString
    - Exceptions (and there are many that you don't see)
  - Program heap layout is affected:
    - Debug heap in OS gets enabled only if launch under debugger
  - GUI Focus changes when breakpoint hit
- Solutions:
  - Attach instead of launch
  - Attach and detach when not required (XP and above)
  - Remote debugging
  - Non invasive WinDbg or CDB (but no break points available)

© Mark Bartosik, www.bugbrowser.com

13

## How do I find a CPU hog?

- What's eating all that CPU time?



© Mark Bartosik, www.bugbrowser.com

©2005 M. Plonsky

14



~\* k

```

Command
0:001> symfix+ c:\websymbols
0:001> .reload /f
0:001> ~* k
~ [threads] * [all] k stack trace

0 Id: 1330.1334 Suspend: 0 Teb: 7f1fd000 Unfrozen
ChildEBP RetAddr
0012fee8 7c90e9c0 ntdll!KiFastSystemCallRet
0012feec 7c8029db ntdll!ZwWaitForSingleObject+0xc
0012ff50 7c802542 kernel32!WaitForSingleObjectEx+0xa8
0012ff64 00401076 kernel32!WaitForSingleObject+0x12
0012ff7c 004011f1 eat_CPU!main+0x46 [b:\talks\accu 2006\debugging talk\eat_cpu\main.cpp @ 33]
0012ffc0 7c81644f eat_CPU!_mainCRTStartup+0x10f [f:\ntm\vc\tools\crt_bld\self_x86\crt\src\cr
0012fffo 00000000 kernel32!BaseProcessStart+0x23

# 1 Id: 1330.13a8 Suspend: 0 Teb: 7f1de000 Unfrozen
ChildEBP RetAddr
0050ffb0 00401015 eat_CPU!do_stuff [b:\talks\accu 2006\debugging talk\eat_cpu\main.cpp @ 6]
0050ffb4 7c80b50b eat_CPU!eat_CPU+0x5 [b:\talks\accu 2006\debugging talk\eat_cpu\main.cpp @ 1]
0050ffec 00000000 kernel32!BaseThreadStart+0x37

2 Id: 1330.12c8 Suspend: 0 Teb: 7f1dd000 Unfrozen
ChildEBP RetAddr
0060ff40 7c90d85c ntdll!KiFastSystemCallRet
0060ff44 7c8023ed ntdll!NtDelayExecution+0xc
0060ff9c 7c802451 kernel32!SleepEx+0x61
0060ffac 00401028 kernel32!Sleep+0xf
0060ffb4 7c80b50b eat_CPU!do_nothing+0x8 [b:\talks\accu 2006\debugging talk\eat_cpu\main.cpp @ 1]
0060ffec 00000000 kernel32!BaseThreadStart+0x37
  
```

In VS 2005 in the command window:>Macros.Samples.VSDebugger.DumpStacks

© Mark Bartosik, www.bugbrowser.com

15

## !runaway

```

Command
0:003> !runaway
User Mode Time
Thread Time
1:10f4 0 days 0:00:15.796
3:1610 0 days 0:00:00.000
2:d80 0 days 0:00:00.000
0:1580 0 days 0:00:00.000
0:003> g
(1138.d64): Break instruction exception - code 80000003 (first chance)
eax=7f1de000 ebx=00000001 ecx=00000002 edx=00000003 esi=00000004 edi=00000005
eip=7c901230 esp=0036ffcc ebp=0036ff14 iopl=0         nv up ei pl zr na po nc
cs=001b  e8=0023  ds=0023  es=0023  fs=0038  gs=0000             efl=00000246
ntdll!DbgBreakPoint:
7c901230 cc                int     3
0:003> !runaway
User Mode Time
Thread Time
1:10f4 0 days 0:00:17.812
3:d64 0 days 0:00:00.000
2:d80 0 days 0:00:00.000
0:1580 0 days 0:00:00.000
0:003> ~* k
ChildEBP RetAddr
0050ffb0 00401015 eat_CPU!do_stuff [b:\talks\accu 2006\debugging talk\eat_cpu\main.cpp @ 6]
0050ffb4 7c80b50b eat_CPU!eat_CPU+0x5 [b:\talks\accu 2006\debugging talk\eat_cpu\main.cpp @ 1]
0050ffec 00000000 kernel32!BaseThreadStart+0x37
  
```

© Mark Bartosik, www.bugbrowser.com

16



TID 16f4 critsec\_a 00405748  
TID 1214 critsec\_b 00405778

## How do I debug a deadlock?

```
0:002> !locks
CritSec dead_lock!critsec_a+0 at 00405748
LockCount          1
RecursionCount     1
OwningThread       16f4
EntryCount         1
ContentionCount    1
*** Locked

CritSec dead_lock!critsec_b+0 at 00405778
LockCount          1
RecursionCount     1
OwningThread       1214
EntryCount         1
ContentionCount    1
*** Locked
```



```
~* k
0:002> ~* kv
0 Id: 1684.af8 Suspend: 2 Tab: 7ffdf000 Unfrozen
ChildEBP RetAddr  Args to Child
0012fee8 7c90e9c0 7c8025db ntdll!KiFastSystemCallRet
0012feec 7c8025db 000017f4 ntdll!ZwWaitForSingleObject
0012ff50 7c802542 000017f4 kernel32!WaitForSingleObjectEx
0012ff64 004011f6 000017f4 kernel32!WaitForSingleObject
0012ff7c 0040182d 00000001 dead_lock!main
0012ffc0 7c816d4f 00000000 dead_lock!__tmainCRTStartup
0012fff0 00000000 00401976 kernel32!BaseProcessStart

1 Id: 1684.16f4 Suspend Unfrozen
ChildEBP RetAddr  Args to Child
0050fef0 7c90e9c0 7c91901b ntdll!KiFastSystemCallRet
0050fef4 7c91901b 000017d8 ntdll!ZwWaitForSingleObject
0050ff7c 7c90104b 00405778 ntdll!RtlpWaitForCriticalSection
0050ff84 0040109d 00405778 ntdll!RtlEnterCriticalSection
0050ff8c 00401130 4ad50ce7 dead_lock!function2
0050ffb4 7c80b50b 00000000 dead_lock!thread_a
0050ffec 00000000 004010d0 kernel32!BaseThreadStart

# 2 Id: 1684.1214 Suspend Unfrozen
ChildEBP RetAddr  Args to Child
0060fedc 7c90e9c0 7c91901b ntdll!KiFastSystemCallRet
0060fee0 7c91901b 000017d8 ntdll!ZwWaitForSingleObject
0060ff68 7c90104b 00405748 ntdll!RtlpWaitForCriticalSection
0060ff70 00401034 00405748 ntdll!RtlEnterCriticalSection
0060ff8c 004011a0 4ae50ce7 dead_lock!function3
0060ffb4 7c80b50b 00000000 dead_lock!thread_b
0060ffec 00000000 00401140 kernel32!BaseThreadStart
```

This thread already owns **critsec\_a** and is attempting to acquire **00405778**

This thread already owns **critsec\_b** and is attempting to acquire **00405748**

## Typical deadlock causes

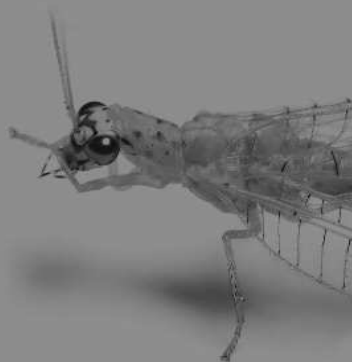
1. “Upward” calls while holding a lock:  
Observer pattern / publish subscribe, or any synchronous generic callback mechanism
2. Order of locking  
thread 1: locks a, b thread 2: locks b, a
3. Lack of RAI
4. Exception with lock acquired (even causing thread to exit)
5. Double acquire on non recursive locks  
e.g. spinlocks, and mutexes
6. Priority inversion – very rare

© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

19

## How do I find memory corruptions?

- Debug build?
- Bounds checker?
- Purify?
- Debugging suit de jour?
- Application verifier, gflags
- Custom allocators  
Using `VirtualAlloc`, `VirtualProtect`,  
and `VirtualFree`
- Data break points
- Custom break point control  
<http://www.morearty.com/code/breakpoint/>



© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

20

# gflags

Part of  
Debugging Tools for Windows

Do not enable both heap tail  
checking and page heap



Global Flags - \*

System Registry | Kernel Flags | Image File | Verifier

Image: (TAB to refresh)

Stop on exception  
 Show loader snaps

Enable heap tail checking  
 Enable heap free checking  
 Enable heap parameter checking  
 Enable heap validation on call

Enable application verifier

Enable heap tagging  
 Create user mode stack trace database

Enable heap tagging by DLL

Load image using large pages if possible  
 Debugger:   
 Stack Backtrace: (Megs)

Disable stack extension  
 Enable system critical breaks  
 Disable heap coalesce on free  
 Enable page heap  
 Early critical section event creation  
 Load DLLs top-down (Win64 only)  
 Disable protected DLL verification

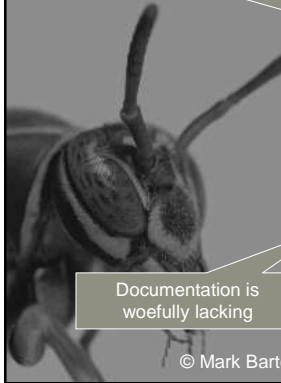
© Mark Bartosik, www.bugbrowser.com

21

# gflags

Part of  
Debugging Tools for Windows

Consider unaligned if  
you can, otherwise  
alignment is 16 bytes



Documentation is  
woefully lacking

Global Flags

System Registry | Kernel Flags | Image File | Verifier

Image: (TAB to refresh)

Enable

PageHeap

Conserve Memory

Size Range Start:  End:   
 Dil Range Start:  End:   
 Random  
 Faults Rate:  Timeout:

OverrunProtection  
 Overrun  Underrun

Dil Names

Unalign  
 Traces  
 No Sync

Handle  
 Stacks

TLS  
 Dirty Stacks  
 RPC Checks  
 COM Checks  
 Propagate Settings  
 Debugger:

Decommit  
 Protect  
 No Lock Checks  
 Dangerous APIs  
 Race Checks  
 Deadlock Checks  
 Virtual Mem  
 Locks  
 First Chance  
 Output Buffer

© Mark Bartosik, www.bugbrowser.com

22

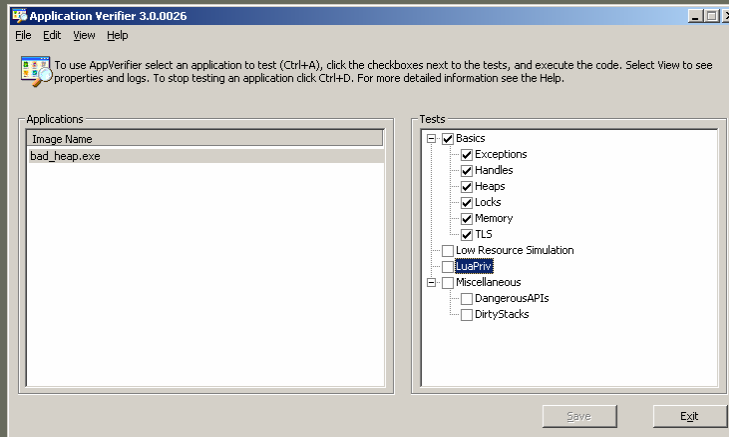
# Application Verifier

Integrated with VS 2005

OR [exclusive or]

Downloadable from [www.microsoft.com](http://www.microsoft.com)

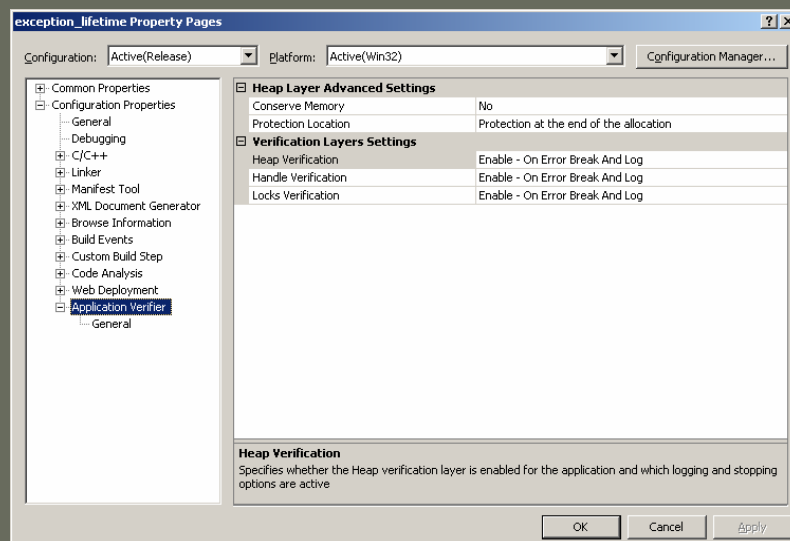
(but not both MSDN Article ID: 911142)



© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

23

# Application Verifier (in VS 2005)



© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

24

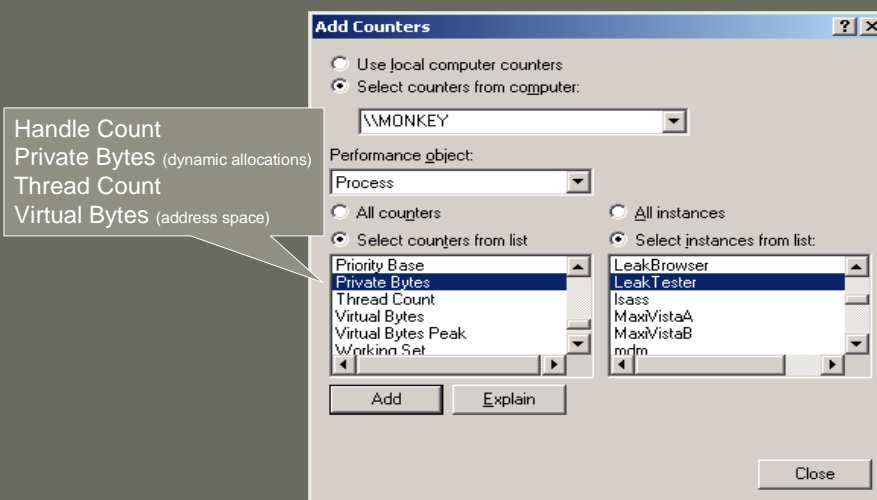
## How do I find leaks?

- Find what's leaking all my memory?
- Leak definition:  
Over time your program continues to consume more resources.
- Perfmon  
Process\Private Bytes (committed memory)  
Process\Handle Count  
Process\Virtual Bytes (address space)

© Mark Bartosik, www.bugbrowser.com

25

## How do I check for leaks?



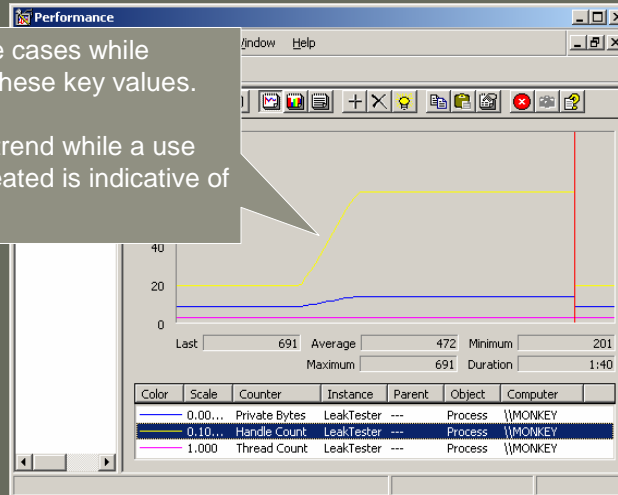
© Mark Bartosik, www.bugbrowser.com

26

## How do I check for leaks?

Perform use cases while monitoring these key values.

An upward trend while a use case is repeated is indicative of a leak.

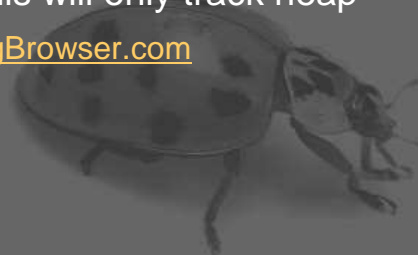


© Mark Bartosik, www.bugbrowser.com

27

## How do I find leaks?

- Memory might not be “garbage”
- Might not be practical to wait for process exit.
- Might not be practical to rebuild everything.
- Program may have many one time only “leaks”
- Microsoft’s UMDH but this will only track heap
- Leak Browser ☺ [www.BugBrowser.com](http://www.BugBrowser.com)

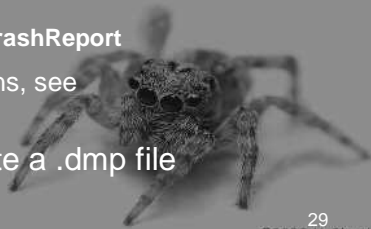


© Mark Bartosik, www.bugbrowser.com

28

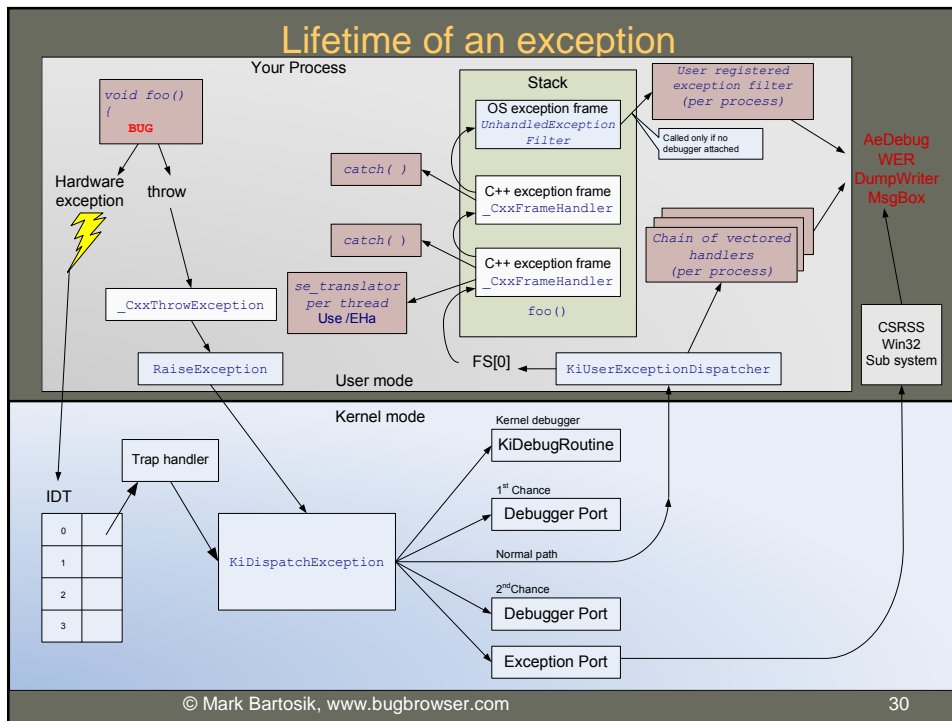
## .dmp files

- Being able to capture an exception, a crash, or broken state in a .dmp file is the sharpest weapon against bugs.
- WinDbg, CDB, AdsPlus  
non redistributable
- Windows Error Reporting (WER)  
Requires WinQual membership, issues with configuration
- DumpWriter  
[www.bugbrowser.com/dumpwriter](http://www.bugbrowser.com/dumpwriter)
- CodeProject, look for CrashRpt and XCrashReport
- Debugging Applications, John Robins, see  
SnapCurrentProcessMiniDump API
- The trick is knowing when to create a .dmp file



© Mark Bartosik, www.bugbrowser.com

29  
©2003 M. Plonsky

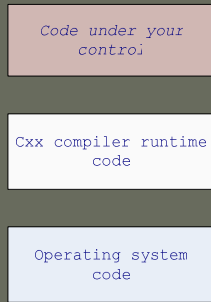


© Mark Bartosik, www.bugbrowser.com

30

## Lifetime of an exception

### Key for exception lifetime

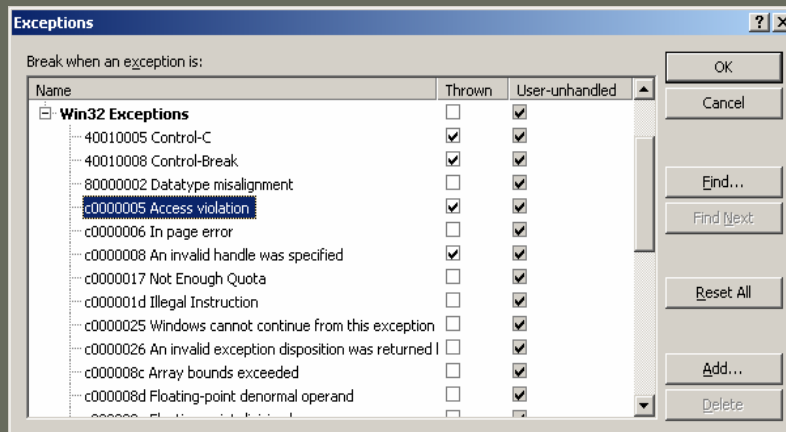


You have the following opportunities to detect a bug and create a .dmp file:

1. Assertions (instead of throw)
2. vectored exception handler
3. se\_translator routine
4. \_\_except
5. catch block
6. unhandled exception filter

## Debugging exceptions

- Always trap access violations
- First column is *first chance* second column is *second chance* (unhandled)





## Magic values

- Values to look for

- 0xFDFDFDFD - No man's land
- 0xDDDDDDDD - freed memory / deleted
- 0xCDCDCDCD - uninitialized
- 0xCCCCCCCC - uninitialized locals
- 0xBAADF00D - value no longer valid
- 0xDEADBEEF - value no longer valid
- ???????? - uncommitted memory

## Capturing exceptions with vectored exception handlers

- Windows XP and beyond  
`_WIN32_WINNT 0x0501`
- `AddVectoredExceptionHandler`
- This installs a **per process** routine that is called whenever a structured exception occurs.
- There is a chain of these routines.
- It is the **first place in user mode code to trap all exceptions**. (First supported place)
- It is suitable for **CONDITIONALLY** creating a dump file.

## Capturing exceptions with se\_translator

- The compiler provides a way to map structured exceptions (like 0xC0000005) to a C++ exception like `std::runtime_error("FATAL!")`.
- `_set_se_translator`
- For VS 7.1 and greater requires `/EHa`, which has more overhead
- Is installed **per thread per CRT**
- Do you have control of all threads?
- Do you have multiple C++ runtimes loaded in your process?
- Can you require `/EHa` ?
- Good for component writers linking with **static CRT**.

© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

35

## Capturing exceptions with unhandled exception filter

- Exceptions often **swallowed by `catch(...)`** before they reach the unhandled exception filter.
- Vulnerable to stack corruption, because it is called at the end of a linked list of filters which is stored on the stack.
- **Per process** filter installed by `SetUnhandledExceptionFilter`
- Trigger your own post mortem or leave it to Microsoft.
- The OS installs a default handler  
Default handler looks as `AeDebug` registry key, runs debugger if configured.  
Win2K: `MessageBox` or `DrWtsn32`  
XP, 2003: if no debugger is installed (or `Drwtsn32`) loads `faultrep.dll` and calls `ReportFault`

© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

36

## Capturing exceptions with catch blocks

- Use `catch ( std::exception & e )` for recovery
- `catch (...)` is evil, but necessary because not all C++ exceptions are rooted in `std::exception`.
- `catch (...)` is evil, because it stops unhandled exception filter from activating.
- Use `catch (...)` for recovery only if you have already trapped fatal exceptions and triggered post a mortem dmp.
- Catch blocks are not good places to trigger a dump because the stack is unwound and destructors have run, possibly destroying evidence.

© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

37

## Capturing exceptions with \_\_try , \_\_except

- Cannot mix `try` and `__try` in the same function
- Cannot use `__try` in functions that require object unwinding
- `GetExceptionInformation` is treated as a keyword
- A `catch(...)` lower in the callstack can swallow exceptions
- Can provide for more localized control, e.g. per thread, per function becomes messy
- Can be messy

Example code:

```
LONG WINAPI MyExceptionFilter(EXCEPTION_POINTERS* ExceptionInfo)
{
    // Decide whether to create a .dmp file
    :
}

void c_plus_plus_function();

void wrapper_around_c_plus_plus_function()
{
    __try
    {
        c_plus_plus_function();
    }
    __except( MyExceptionFilter( GetExceptionInformation() ) )
    {}
}
```

© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

38

## Capturing post mortem files programmatically

- Implement a **vectored exception handler**, not simple, catches almost everything.
- Implement an **unhandled exception filter**, easy but does not catch everything.
- Implement an **se\_translator** function easy but not suitable for all projects.
- Implement an exception filter with **\_\_except**, can be messy, scoped, easy for per thread, does not catch everything.
- Implement `catch(...)`, not recommended.
- Whatever the mechanism used to intercept exceptions, we need a process to create the dumpfile (doing this in-proc is not recommended).
- A typical implementation will `CreateProcess` with a command line of "dumpcapture-program -p *pid*".

## Capturing post mortem files

- **PER MACHINE** configuration. Controlled by machine administrator. The OS installed unhandled exception filter looks at the `AeDebug` registry key will automatically launch the debugger registered by this key. This is done by an internal kernel32 function called `UnhandledExceptionFilter`.

HKEY\_LOCAL\_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\AeDebug

Values are `Auto` (DWORD set to 1)

and `Debugger` (String command line for debugger)

- `drwtsn32 -p %ld` (will use WER instead under XP and later)
- `patho\ntsd -p %ld -e %ld -g -c ".dump /ma /u c:\TEMP\new.dmp; q"`
- `patho\cdb -p %ld -e %ld -g -c ".dump /ma /u c:\TEMP\new.dmp; q"`
- `patho\windbg -p %ld -e %ld -g -c ".dump /ma /u c:\TEMP\new.dmp; q"`
- `patho\dumpwriter -p %ld -e %ld`

- **PER APPLICATION** configuration. Controlled by application developer.
  - Call `CreateProcess` or `system` or similar to launch a process like those above. This can be done from within your chosen exception interception routine, you can control where dump file is written and how much information is included. This gives the developer much finer grain control.
  - Call `ReportFault`. This reports to Microsoft via WER (XP and above).

## Configuring DrWtsn32 Dr. Watson

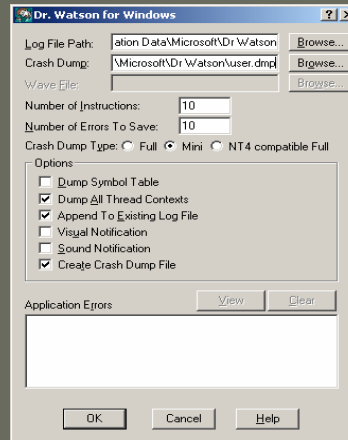
### DrWatson has retired!

(Still used on NT4 and 2000)

DrWtsn32 /?  
to configure

DrWtsn32 -p pid  
To create a dump file,  
and add to the logs,  
but only ONE dump file is created.  
Not recommended.

DrWtsn32 -i  
To install a post mortem debugger.  
Not recommended



© Mark Bartosik, www.bugbrowser.com

41

## Configuring DumpWriter <http://www.bugbrowser.com/dumpwriter>

- Dumpwriter /?
- Can configure solely via command line
- Can configure with a per installation XML file
- Can configure with a per application XML file
- Can configure with xpath within an XML file
- See HTML documentation
- Open source

© Mark Bartosik, www.bugbrowser.com

42

## Using WER

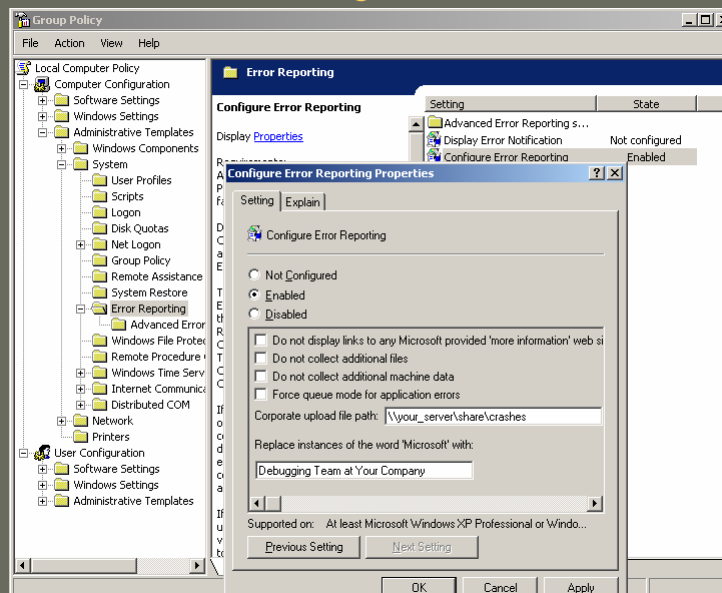
<https://winqual.microsoft.com> must use IE

- WER is configured via Group Policy editor `GPEDIT.MSC`  
Path in GPEDIT is `\Computer Configuration\Administrative Templates\System>Error Reporting`
  - Enable / disable.
  - Enable a corporate upload path (UNC path)  
This can be a network path for your test lab, or a local path.
  - Cannot redirect to a different web address ☹
- Basic requirements
  - Class 3 Verisign code signing certificate (\$500).
  - Free once you have the Verisign cert.  
(billing is only for other WinQual services, not WER)
  - Sign the legal agreement (privacy of user data etc.)
  - You submit your products' released modules to Microsoft to create a mapping.  
(1 business day to process, 1 week to be reflected on web site planned to reduce 24 hours).
- You can download .cab files with the .dmp files and a little more info.
- Currently have to have 3 crash reports per bucket to display crash reports (a bucket is a crash at a unique RVA in a module).  
(Hope to use stripped symbols to allow buckets to use function names rather than RVA)

© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

43

## Using WER



© Mark Bartosik, [www.bugbrowser.com](http://www.bugbrowser.com)

44

## Using WER

Windows Quality Online Services: Windows Error Reports - Microsoft Internet Explorer

Address: https://winqual.microsoft.com/member/lver/default.aspx

Windows

Winqual Home

Windows Error Reports

- Home
- Kernel Mode
- User Mode

Driver Distribution Center

- Home

Submissions

- Create
- View & Manage

Fabrikam Inc

User Mode Crashes [View Bucket List](#)

Reported Crashes

Bucket ID	Reported Crashes
223351893	20
223351928	15
223351911	15
223640677	10
223640645	10
223132857	8
223640636	7
221286851	6
223614686	5
224199610	5

Top 10 Buckets - Company-wide

- User Mode Home
- View Mapped Files
- Company Summary
- Product Rollups
- Create a New Response
- View/Edit Your Responses
- Learn About Responses

Kernel Mode Crashes [View Bucket List](#)

90 Day Crash Report

Bucket ID	90 Day Crash Report
4619643	12
4635451	6
4734718	1
4983827	1

Top 10 Buckets - Company-wide

- Kernel Mode Home
- Manage Driver Mapping
- My Crashes by Driver
- My Crashes by Device

© Mark Bartosik, www.bugbrowser.com

45

## Using WER

Windows Quality Online Services: Software Crashes - Microsoft Internet Explorer

Address: https://winqual.microsoft.com/member/lver/User/ManageResponsePage.aspx

Windows

Winqual Home

Windows Error Reports

- Home
- Kernel Mode
- Software Home

Crash Data

- Show company summary
- Show product rollups

Customer Responses

- Create a new response
- View / Edit your responses
- Show all buckets with live responses

File Mappings

- List file mappings
- Create new file mappings
- Remove file mappings

Find Buckets

Find in:

- Product name
- Bucket ID
- Executable name
- Module name

[Wildcard Help](#)

[Windows Error Reports Home](#) > [Software Crashes](#) > Manage Response Request

### Create a Response Request

File Name:  Example: Word.exe

Product Name:  Example: Microsoft Office 2003

File Version Range:

Min Version:     Example: 1.0.23.101 (greater than or equal to)

Max Version:     Example: 4.4.65.405 (less than)

Response URL:  Example: http://www.microsoft.com/fixExe

Response Language Mask:  English Only

Response Template:  Product Upgrade

Sample text for selected Template

An Update Is Available  
Thank you for submitting an error report. The error was likely caused by:  
[ProductName] ([FileName]) [ProductName] ([FileName]) was created by:

© Mark Bartosik, www.bugbrowser.com

46

## Using WER (example crash)

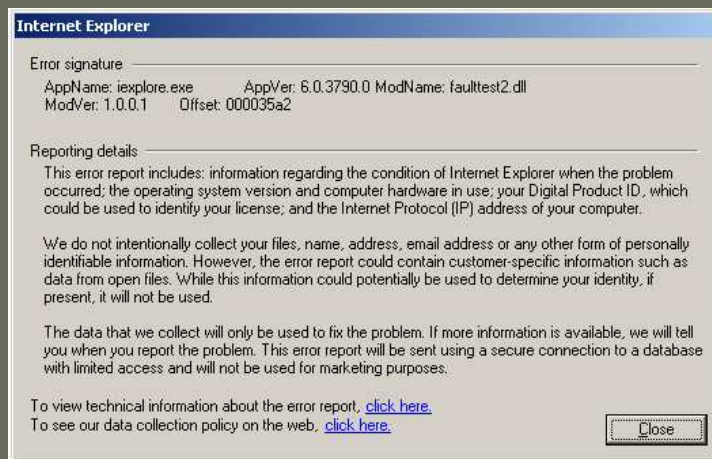
- WER will report Shell detected hangs “Application not responding”
- WER will report unhandled exceptions



© Mark Bartosik, www.bugbrowser.com

47

## Using WER (example crash)

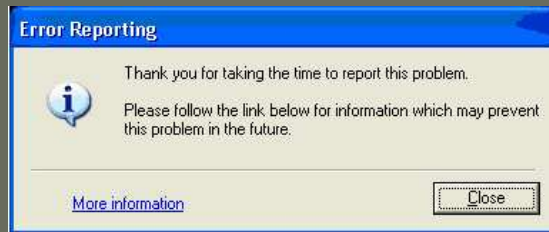


© Mark Bartosik, www.bugbrowser.com

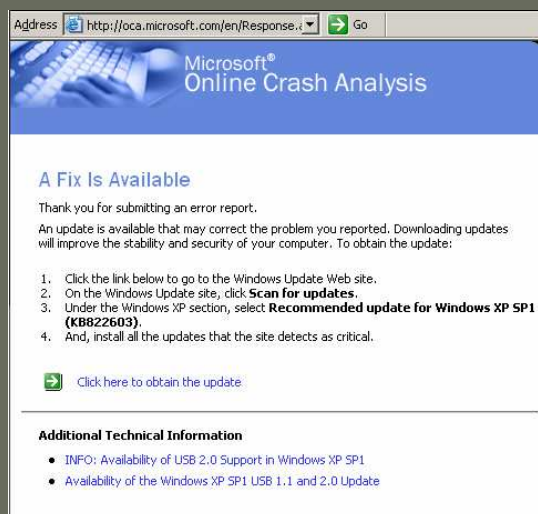
48



## Using WER (example crash)

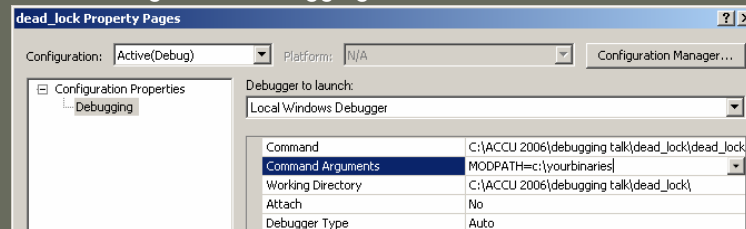


## Using WER (example crash)



## Reading .dmp files with VS 2005

- View the dump file in both WinDbg and Visual Studio.
- VS 2005:
  - Open the .dmp file using File, Open, Project/Solution
  - Set MODPATH to point to where binaries are
  - Set symbol path to include Microsoft symbol server
  - Select Debug, Start Debugging

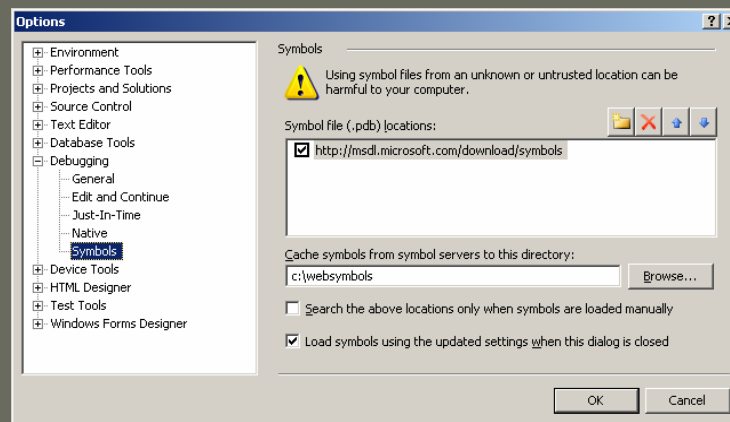


© Mark Bartosik, www.bugbrowser.com

51

## Reading .dmp files with VS 2005

- Always include the Microsoft symbol server on the path (Menu option: Tools, Options)
- Or set `_NT_SYMBOL_PATH` environment variable to include `SRV*c:\websymbols*http://msdl.microsoft.com/download/symbols`



© Mark Bartosik, www.bugbrowser.com

52

## Reading .dmp files with WinDebug

- First download the latest Debugging Tools for Windows package. It is updated about twice a year, plus beta releases, currently at v6.6.
- Set the symbol path:
  - `.sympath`
  - or from the menu*
  - File, Symbol File Path...
- Make sure the Microsoft symbol server is on the path:
  - Add SRV\*c:\websymbols\*http://msdl.microsoft.com/download/symbols
  - or use*
  - `.symfix+ c:\websymbols`
- Make sure that path to the binary images is set
  - `.exepath+ c:\where_your_binaries_are`
  - or from the menu*
  - File, Image File Path...
- File, Open Crash Dump... (older versions also Debug, Go)

## Summary notes (this slide not on ACCU CD)

- Always build with symbols, and archive symbols with .exe (and to your symbol server)
- Use Microsoft symbol server, for large projects maybe setup your own symbol server.
- Use source server (integrate it with your build, read more in Debugging Tools for Windows help file)
- Sign up for WER is you can, if not use dumpwriter from [www.bugbrowser.com](http://www.bugbrowser.com)
- Unoptimized builds really help debugging
- Capture a .DMP file if you can, either using WER (can confirm local path), or dumpwriter or similar or windbg or your debugger
- Use VS 2005 debugger (and WinDbg – but Windbg is not user friendly)
- How to debug high CPU usage
- How to debug deadlocks
- How to debug memory overwrites
- Globals are handy for debugging (but bad for design usually)
- Enable trap on Access Violation in the exceptions dialog of Visual Studio.
- Several ways to trap an exception to generate a .dmp file (best is vectored exception handler)

# Links and references

## Links

- Latest slides, DumpWriter, Leak Browser: [www.bugbrowser.com](http://www.bugbrowser.com)
- Insect photos thanks to: [www.mplonsky.com](http://www.mplonsky.com)
- WinDbg help: [news://microsoft.public.windbg](http://news://microsoft.public.windbg)
- Debugging Tools for Windows: <http://www.microsoft.com/whdc/devtools/debugging/default.msp>
- Programmed breakpoint control: <http://www.morearty.com/code/breakpoint>
- Windows Error Reporting (WER):
  - <http://wingual.microsoft.com> (must use IE for this site)
  - <http://microsoft.sitestream.com/PDC05/FUN/FUN313.zip>
  - [http://microsoft.sitestream.com/PDC05/FUN/FUN313\\_files/Botto\\_files/FUN313\\_Hardest.ppt](http://microsoft.sitestream.com/PDC05/FUN/FUN313_files/Botto_files/FUN313_Hardest.ppt)
- Code Project:
  - [http://www.codeproject.com/debug/postmortemdebug\\_standalone1.asp](http://www.codeproject.com/debug/postmortemdebug_standalone1.asp)
  - [http://www.codeproject.com/debug/crash\\_report.asp](http://www.codeproject.com/debug/crash_report.asp)
  - <http://www.codeproject.com/debug/XCrashReportPt1.asp>

## Books

- Debugging Applications for Microsoft .NET and Microsoft Windows, ISBN:0735615365  
(I disagree with a lot of advice in this book – like recommendations not to use STL and even some debugging aspects, nevertheless it presents useful techniques)
- Microsoft Windows Internals, ISBN:0735619174 (currently 4<sup>th</sup> Edition)
- How Debuggers Work, ISBN 0-471-14966-8