FREE eBook

LEARNING WinDbg

Free unaffiliated eBook created from
Stack Overflow contributors.

#windbg

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: windbg

It is an unofficial and free WinDbg ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official WinDbg.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with WinDbg

Remarks

This section provides an overview of what windbg is, and why a developer might want to use it.

It should also mention any large subjects within windbg, and link out to the related topics. Since the Documentation for windbg is new, you may need to create initial versions of those related topics.

Versions

Important versions of WinDbg, for supported versions of WinDbg. See also a detailed list with historical versions online.

It's important to note that there's a versioning scheme change from older 6.12 to the newer 6.1 version. The older versions have low numbers (<100) in the third place while newer versions have high numbers (>6000).

In many cases, WinDbg versions provided for newer Windows versions still work on older versions on Windows, e.g. Version 10 of WinDbg can still be used on Windows 7. However, some commands may make use of API calls that are not available and thus fail. Therefore it's good to have several versions of WinDbg available.

Version	Description	Release Date
6.12.0002.633	provided for Windows 7 and .NET Framework 4	2010-05-21
6.1.7600.16385		2009-07-24
6.2.8400.0	update for Windows 8 (?)	2012-06-23
6.2.9200.16384	provided for Windows 8 and .NET Framework 4.5	2012-11-15
6.3.9600.16384	provided for Windows 8.1	2013-10-17
10.0.10075.9	provided for Windows 10	2015-04-29
10.0.10586.567	provided since Windows 10, build 1511	2015-10-30
10.0.14321.1024	provided since Windows 10, build 1607	2016-07-29

Examples

Installation or Setup

Microsoft describes 3 ways of installing WinDbg:

- as part of the WDK (Windows Driver Kit)
- as part of the SDK (Software Development Kit)
- with the installer of the SDK and deselecting everything else but "Debugging Tools for Windows"

To get the installer, visit Download the WDK, WinDbg, and associated tools and scroll down to a section called "Get debugging tools".

A well-known and convenient but inofficial source is Codemachine where you can also download older versions of the Debugging Tools directly.

The setup itself is straight-forward. Click through the installer until it finishes.

Debuggers

WinDbg is often used as an abbreviation of "Debugging tools for Windows". It contains different debuggers:

Debugger	Description
WinDbg	the debugger with a graphical user interface
CDB	**c**onsole **d**e**b**ugger, user mode debugger which runs in the currently open console
NTSD	**n**ew **t**erminal **s**ymbolic **d**ebugger, user mode debugger which opens a new terminal (console) as the name suggests
KD	the **k**ernel **d**ebugger, which runs in the currrently open console
NTKD	**n**ew **t**erminal **k**ernel **d**ebugger, opens a new terminal

The commands are identical, except that there may be GUI related commands which don't work in the console versions.

Read Getting started with WinDbg online: https://riptutorial.com/windbg/topic/1833/getting-started-with-windbg

Chapter 2: Crash analysis

Examples

Basic user mode crash analysis

`.exr -1` gives you details about the last exception thrown.

`!analyze -v` usually does a good job as well.

For .NET, the command `!pe` of the SOS extension shows details about the .NET exception that was thrown.

Read Crash analysis online: https://riptutorial.com/windbg/topic/5389/crash-analysis

Chapter 3: DML(Debugger Mark Language)

Examples

Turn on/off

.prefer_dml 1 turn on dmlformat output

.prefer_dml 0 turn off dmlformat output

Read DML(Debugger Mark Language) online: https://riptutorial.com/windbg/topic/7987/dml-debugger-mark-language-

Chapter 4: Extensions

Examples

SOS

SOS (son of strike) is the official WinDbg extension from Microsoft for .NET. It gets installed as part of the .NET framework and thus is available by default.

Like any extension, it can be loaded using `.load x:\full\path\to\sos.dll`, but there are easier ways. Depending on the version of .NET, the extension is located side by side to `mscorwks.dll` (.NET CLR 2), `clr.dll` (.NET CLR 4) or `coreclr.dll` (Silverlight and Universal apps), so one of the following commands should work:

```
.loadby sos clr
.loadby sos coreclr
.loadby sos mscorwks
```

For a list of available commands, consult `!help`.

SOSex

SOSex is an extension to SOS, written by Steve Johnson, a Microsoft employee. He provides SOSex for download for free, but it's not open source.

Typically, the extension is not available side by side to any other DLL, so it is usually loaded with `.load x:\full\path\to\sosex.dll`.

Besides simplifying debugging of .NET, the command `!dlk` can also be used in native environments for checking deadlocks of critical sections.

For a list of available commands, consult `!help` of SOSex.

PyKD

PyKD is a WinDbg extension that enables you writing Python scripts. It's open source.

Typically, the extension is not available side by side to any other DLL, so it is usually loaded with `.load x:\full\path\to\pykd.pyd`, where PYD is the extension for a python DLL, but you can rename it to DLL if you like.

Getting started with PyKd

PyKD does not offer `!help`, so look up the documentation at Codeplex. Many developers seem to be from Russia and the most up-to-date and complete documentation is probably in Russian. The Google translater does a decent job.

Like other extensions, use the correct bitness of the extension that corresponds to that of WinDbg. In addition to that you must have Python installed with the same bitness as well.

`!py` runs an REPL interpreter and `!py x:\path\to\script.py` runs a python script. Scripts should use

```
from pykd import *
```

as the first line in order to make use of PyKD's functionality, while this line is not needed in the REPL interpreter. The interpreter can be exited using `exit()`.

NetExt

NetExt is an extension for .NET which provides

- LINQ-like queries for objects on the heap (`!wselect`, `!wfrom`)
- display capabilities for special objects like dictionaries and hash tables (`!wdict`, `!whash`)
- ASP.NET / HTTP related commands (`!wcookie`, `!wruntime`, `!whttp`)
- several other network related commands

Typically, the extension is not available side by side to any other DLL, so it is usually loaded with .load x:\full\path\to\netext.dll

Extensions overview

An incomplete list of WinDbg extensions that are not installed with WinDbg itself:

Extension	Purpose
SOS	.NET (official Microsoft extension)
SOSex	.NET (extension for SOS)
CoSOS	.NET (extension for SOS)
NetExt	.NET (with focus on networking)
PyKD	Python scripting
PDE	Windows native and store applications (stowed exceptions)
PSSCOR	.NET
SDbgExt	.NET
MEX	.NET

CoSOS

CoSOS (cousin of SOS) is an open source extension for WinDbg focusing on .NET memory fragmentation (`!gcview`) and threading issues (`!wfo`, `!tn`).

Typically, the extension is not available side by side to any other DLL, so it is usually loaded with `.load x:\full\path\to\cosos.dll`. It requires that SOS is loaded and currently works with 32 bit applications only.

Read Extensions online: https://riptutorial.com/windbg/topic/5391/extensions

Chapter 5: Kernel debugging

Examples

Important commands

- !process - list user mode processes
- .process - set process context
- !peb - show process environment block
- !teb - show thread environment block
- !locks - deadlock analysis
- .dump - save a crash dump file to disk

Read Kernel debugging online: https://riptutorial.com/windbg/topic/6076/kernel-debugging

Chapter 6: Remote debugging

Examples

Important commands

- .server - create a debugging server
- .clients - list debugging clients connected to the server
- .endsrv - end a debugging server
- .servers - list debugging server connections
- .remote - start a remote.exe server
- .noshell - prevent shell commands

Read Remote debugging online: https://riptutorial.com/windbg/topic/5977/remote-debugging

Chapter 7: User mode / application debugging

Examples

Important commands

Documenting your work

Remember what you've done and retain long outputs which can't be kept in WinDbg's buffer. It's always good to have a log available for reproducing debugging steps, e.g. to ask questions on Stack Overflow.

Command	Purpose
`.logopen`	create a log file
`.logclose`	close the log file
`.dump`	save crash dump file (snapshot of the current debugging session)

Working with symbols

Without or with incorrect symbols, you may receive wrong information and be misled. Make sure you're familiar with these commands before starting work in WinDbg. See also How to set up symbols in WinDbg.

Command	Purpose
`.symfix`	set or add symbols to official Microsoft symbol path
`.sympath`	set or add own or 3rd party symbols
`.reload`	reload symbols
`.symopt`	define symbol handling options
`!sym`	control symbol loading
`x`	examine symbols
`ln`	list nearest symbols

Crash analysis

Find out what has happened (in crash dumps) and how to handle events (in live debugging).

Command	Purpose
`.exr`	display exception record
`.lastevent`	display last event
`sx`	define exception handling
`!analyze`	analyze a crash or hang
`!avrf`	application verifier

The environment

Check the process name and version information.

Command	Purpose
| (pipe)	process information
`lm`	module list

Threads, call stacks, registers and memory

Inspect the details.

Command	Purpose
`~`	thread list
`r`	registers
`k`	call stack
`d`*	display memory
`e`*	edit memory
`s`	search memory
`.formats`	convert between number formats
`?`	evaluate expression

Command	Purpose
`u*`	disassemble
`a`	assemble
`!address`	memory info

Controlling the target

In live debugging, take control the execution.

Command	Purpose
`g`	go / continue
`gu`	go up
`p`	single step
`t`	trace (single step and output registers)
`bp`	set breakpoint
`bl`	breakpoint list

Working with extensions

Extensions may provide significant advantages and enhancements.

Command	Purpose
`.load`	load extension (full path)
`.loadby`	load extension relative to module
`.chain`	display loaded extensions
`.unload`	unload extension

Stop debugging

Command	Purpose
`q`	quit and terminate application
`qd`	detach and quit

Attach and detach

Command	Purpose
`.tlist`	process list
`.attach`	attach to process
`.create`	create a process and attach
`.childdbg`	define child process debugging behavior
`.detach`	detach from a process
`.kill`	kill a process
`.restart`	restart the process

Behavior of WinDbg

Command	Purpose
`.prefer_dml`	set debugger markup language handling
`.effmach`	switch the bitness

Usability Commands

Command	Purpose
`.cmdtree`	Loads a text file with predefined commands in a separate window

Getting Helps

Command	Purpose
`.hh`	Displays the help manual for WinDbg commands

Create Custom Command Window in Windbg

The `.cmdtree` command allows to open a `.txt` file with predefined commands which you can simply double click to execute.

How to create command file

Create the file using this template

```
windbg ANSI Command Tree 1.0
title {"Window title"}
body
{"Group Heading"}
 {"Name of command to display"} {"command"}
 {"Name of command to display"} {"command"}
{"Group Heading"}
 {"Name of command to display"} {"command"}
```

Things to take care

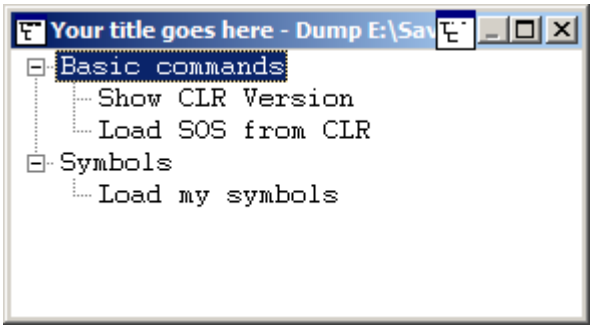
1. The template format should be followed precisely for opening the file in Windbg.
2. The newline is required after each `{Group Heading}`.
3. Each `{Name of command to display} {command}` pair should be in one line and should be followed by a new line.

Example of custom command file

```
windbg ANSI Command Tree 1.0
title {"Your title goes here"}
body
{"Basic commands"}
 {"Show CLR Version"} {"lmv m clr"}
 {"Load SOS from CLR"} {".loadby sos clr "}
{"Symbols"}
 {"Load my symbols"} {".sympath+ "c:\DebugSymbols" ; .reload"}
```

How to open command UI from command window

Execute `.cmdtree <path of your .txt file>` to open the window. You will see a window like this



Double click on the command to execute.

Read User mode / application debugging online: https://riptutorial.com/windbg/topic/5384/user-mode---application-debugging

Credits

S. No	Chapters	Contributors
1	Getting started with WinDbg	Community, Thomas Weller
2	Crash analysis	Thomas Weller
3	DML(Debugger Mark Language)	Wang Zhengzhang
4	Extensions	Jason Evans, Lieven Keersmaekers, Thomas Weller
5	Kernel debugging	Thomas Weller
6	Remote debugging	Thomas Weller
7	User mode / application debugging	Piyush Parashar, Thomas Weller, X. Liu