

WinDbg Cheat Sheet (user mode only)

Help Commands	
?	Help on Debuggee commands
.help	Help on Debugger commands
.hh <i>command</i>	Open WinDbg's help for this command

Execution Control	
restart	Stop and restart execution
t (F11)	Step into (trace)
p [count] (F10)	Step over
pa <i>address</i>	Run to address
pt	Execute until a return instruction is reached
ph [count]	Execute until a branching instruction is reached. count = # of branches reached until it stops
g (F5)	Continue (go)
gu (Shift-F11)	Execute until the current function is complete
(Ctrl-Break)	Break

Breakpoints	
bl	List breakpoints
bp [addr] [" <i>script</i> "]	Set a breakpoint
bp	Set breakpoint at current instruction
bp <i>addr</i>	Set breakpoint at specified address
bp <i>addr</i> " <i>script</i> "	Set a breakpoint and run script when hit bp 403250 ".echo BP hit;g"
bu <i>symbol</i>	Set unresolved breakpoint on a symbol
bm <i>pattern</i>	Set breakpoint on all symbols (unresolved by default) matching the specified pattern
bm /d <i>pattern</i>	Converts the breakpoints to addresses
bc #	Clear a breakpoint
bc *	Clear all breakpoints
bd #	Disable a breakpoint
bd *	Disable all breakpoints
be #	Enable a breakpoint
be *	Enable all breakpoints
ba [access] [size] <i>addr</i>	Set a breakpoint on memory access Size can be 1, 2, or 4 Access: r = Break on read access w = Break on write access e = Break on execute access

Listing Modules	
lm [olfv]	List all modules
lm o	List only loaded modules
lm l	List modules with symbol information
lm f	List all modules and their full image path
lm v	List all modules and be verbose
lm a <i>address</i>	Display the module that contains <i>address</i>
lm m <i>pattern</i>	Find module name, can contain wildcard
lm M <i>pattern</i>	Find image path, can contain wildcard

Symbols	
.reload /f	Reload all symbols
ld <i>module</i>	Load symbols for a module
ld *	Load symbols for all modules
ln <i>address</i>	Find nearest symbol to address
x <i>module!</i> <i>symbol</i>	Display the symbols that match the specified pattern, can contain wildcard

Registers	
r	Display all registers and their values
r <i>reg</i>	Display a single register and its value
r <i>reg=value</i>	Set the register to a specific value

Unassembly	
u[ub] <i>address</i> [L#]	Unassemble from memory
uu <i>address</i>	Disassembly continues past read error
ub <i>address</i>	Determine range by counting backwards
u <i>address</i> L#	Set the number of instructions to disassemble

Display Memory	
d{ <i>type</i> } [/c#] <i>addr</i> [L#]	Display the contents of memory. Types: b = Bytes + ASCII characters w = WORD (2 bytes) W = WORD + ASCII characters d = DWORD (4 bytes) c = DWORD + ASCII characters q = QWORD (8 bytes) a = ASCII string up until first null byte u = Unicode string up until first null byte f = Single-precision float numbers (4 bytes) D = Double-precision float numbers (8 bytes)
d* /c# <i>addr</i>	Set the number of columns to use in the display
d* <i>addr</i> L#	Set the length of output

Searching Memory	
s -[<i>type</i>] <i>range pattern</i>	Search memory range for pattern. Types: b = Byte w = WORD (2 bytes) d = DWORD (4 bytes) q = QWORD (8 bytes) a = ASCII string u = Unicode string
s -a 0 L?80000000 " <i>string</i> "	Search entire user process memory space for a string. Must use "L?" if range is > 256 MB.
s -d 0 L?80000000 41414141	Search entire user process memory space for a DWORD value.

Display Type Information	
dt [-r] <i>name</i>	Display variable or data type information
dt -r <i>name</i>	Recursively dump the subtype fields
dt <i>name addr</i>	Specify the address of the struct
dt ntdll!_TEB @\$teb	Use @ to specify a register
dt <i>name field</i>	Specify the field to display

Display Memory & Symbols	
dds <i>range</i>	Display DWORD (4 byte) values & symbols
dqs <i>range</i>	Display QWORD (8 byte) values & symbols
dps <i>range</i>	Display pointer-sized (4 or 8 byte) values & symbols

Evaluate Expressions	
? <i>expr</i>	Evaluates an expression. Examples: ? 77269bc0 - 77231430 ? 77269bc0 >> 18 ? 41 (to see value in decimal)
?? <i>expr</i>	Evaluates C++ expression. Example: ?? sizeof(ntdll!_TEB)
.formats <i>expr</i>	Evaluate and show in multiple formats

Prefixes	
0x	Hexadecimal (default)
0n	Decimal
0y	Binary

Miscellaneous	
k[bpPv]	Display stack backtrace p = Display all parameters passed to each function P = Same as p but printed on a second line
a [<i>address</i>]	Assemble x86 instructions and puts the resulting op codes into memory. <i>address</i> specifies the start of memory where the resulting codes are put.

Extensions	
! <i>address</i> [<i>address</i>]	Display a memory map. <i>address</i> specifies the address for the region to display.
!exchain	Display the current exception handler chain
!vprot <i>address</i>	Display virtual memory protection information
!dh <i>address</i>	Display the headers for the specified image