Thread Navigation

~ List thread map
~#s Set thread context to #

- indicates exception thread
. – indicates active thread

Example:
~3s – change to thread 3

Frame Navigation

.frame – display current frame
.frame # - change frame to #

Example:
.frame 9 – change to frame 9

Display Memory

dd – display dword
dc – display dwords and ASCII
da – display ASCII
du – display Unicode
poi – dereference pointer (use with d* commands

Examples:
dd 0x12345678 – display val
dd poi (0x12345678) – dereference pointer and display contents at address 0x12345678

Stack Trace

k – stack trace
n – add frame numbers
v – include FPO data
b – display 1st three params
L – suppress source line #s
p – display full parameters
P – display full parameters on separate line
f – show distance between stack frames

Examples:
~3kn – display thread 3 stack with frame numbers
~*k – display all stacks

Disassembly

u – unassembled
uf – unassembled function

[ebp-0x4] – 1st local var
[ebp-0x8] – 2nd local var
[ebp-0xc] – 3rd local var
[ebp-0x#] - #/4 th local var
[ebp+0x4] – return addr of caller
[ebp+0x8] – 1st parameter
[ebp+0xc] – 2nd parameter
[ebp+0x10] – 3rd parameter
[ebp+0x#] - #/4 th parameter

ebp – frame pointer, stays constant during function
esp – stack pointer, points to top of stack; changes with commands like push, sub, add
eip – instruction pointer, points to current instruction

Prolog/Epilog

Prolog – start of function
```
push ebp
mov ebp, esp
sub esp, #  (# bytes locals)
```

Epilog – end of function
```
mov esp, ebp
pop ebp
ret
```

Never examine stack or local variables during prolog or epilog because ESP and EBP inconsistent

Display Data and Symbol

dds – dump dword and symbol

Mainly used for raw stack dump

Example:
dds esp – dump stack and display symbols
dds esp L32 – dump stack and display 50 entries

Local Vars and Types

dv – display local variables
dt – display type
dt –r display type recursively

Must set thread and frame context first

Examples:
~2s;.frame 3;dv
change context to thread #2, set frame to #3, and display local vars

Registers

r – display all registers
r @eax – display EAX register
r @eax = value
-- set EAX to value

Example:
r @eip = 0x00123456
set EIP to 0x00123456

General Purpose:
EAX – accumulator
EBX – base
ECX – count
EDX – double-precision
ESI – source index
EDI – destination index
EIP – Instruction pointer
EBP – base frame pointer
ESP – Stack pointer

ESP, EBP, EIP manage code position and stack details

EAX, EBX, ECX, EDX – although have meanings, are used for data manipulation

ESI, EDI – used for source pointers and many string commands such as SCAS, STOS

Breakpoints

bp {address | symbol}
– sets breakpoint on either address (0x#) or symbol
bl – list breakpoints
bc # - clear breakpoint #
bd # - disable breakpoint
be # - enable breakpoint
bp address "command"

Examples:
bp 0x00123456
-- set breakpoint at address 0x00123456
bp ntdll!malloc
-- set breakpoint at symbol
bc 3
-- clear breakpoint #3
bc 4,7
-- clear breakpoints 4 and 7
bc *
-- clear all breakpoints
bp 0x00123456 "kvn;g"
-- break and dump stack and go automatically

Symbol Loading

.symfix+ c:\symbols
-- adds Microsoft symbol server to path
ld module
-- force load module symbols
lm – list modules
lmlv – list modules list version

- Separate paths with ; character
- Always click Reload check box after altering symbol path

Example:
ld ntdll
-- force loads ntdll.pdb

Critical Sections

!cs – list all critical sections
!cs –l -- list locked critical sections
!cs address -- list critical section at address

Debug Display

.echo sometext
--- prints "sometext" in the command output window
.echotimestamps
--- prints date/time of event

Log File

.logopen filename
-- opens file "filename" for logging
.logclose
-- closes current log file

Search Memory

s-a addr1 addr2 text
-- search from addr1 to addr2 for 'text' in ASCII
s-u addr1 addr2 text
-- search from addr1 to addr2 for 'text' in Unicode
s-[w]a addr1 addr2 text
-- search from addr1 to addr2 for writable 'text' in ASCII
s-[w]u addr1 addr2 text
-- search from addr1 to addr2 for writable 'text' in Unicode
s-{a|u} startaddr L{size} text
-- search for text starting at 0xstartaddr and search through 'size' addresses

---

Examples:
s-a 0012fa00 0012fadc hello
-- finds string 'hello' in the specified range
s-u 0012fa00 0012fadc hello
-- finds string 'h.e.l.l.o.' in the specified range

Note: the dots in 'h.e.l.l.o.' just represent Unicode, not a 'dot'