NAME

winedbg – Wine debugger

SYNOPSIS

winedbg [*options*] [*program_name* [*program_arguments*] | *wpid*]

winedbg --gdb [*options*] [*program_name* [*program_arguments*] | *wpid*]

winedbg --auto *wpid*

winedbg --minidump [*file.mdmp*] *wpid*

winedbg *file.mdmp*

DESCRIPTION

winedbg is a debugger for Wine. It allows:

+ debugging native Win32 applications
+ debugging Winelib applications
+ being a drop-in replacement for Dr Watson

MODES

winedbg can be used in five modes. The first argument to the program determines the mode winedbg will run in.

default Without any explicit mode, this is standard **winedbg** operating mode. **winedbg** will act as the front end for the user.

--gdb **winedbg** will be used as a proxy for **gdb**. **gdb** will be the front end for command handling, and **winedbg** will proxy all debugging requests from **gdb** to the Win32 APIs.

--auto This mode is used when **winedbg** is set up in *AeDebug* registry entry as the default debugger. **winedbg** will then display basic information about a crash. This is useful for users who don't want to debug a crash, but rather gather relevant information about the crash to be sent to developers.

--minidump

This mode is similar to the **--auto** one, except that instead of printing the information on the screen (as **--auto** does), it's saved into a minidump file. The name of the file is either passed on the command line, or generated by **WineDbg** when none is given. This file could later on be reloaded into **winedbg** for further examination.

file.mdmp

In this mode **winedbg** reloads the state of a debuggee which has been saved into a minidump file. See either the **minidump** command below, or the **--minidump mode**.

OPTIONS

When in **default** mode, the following options are available:

--command *string*

winedbg will execute the command *string* as if it was keyed on winedbg command line, and then will exit. This can be handy for getting the pid of running processes (winedbg --command "info proc").

--file *filename*

winedbg will execute the list of commands contained in file filename as if they were keyed on winedbg command line, and then will exit.

When in **gdb** proxy mode, the following options are available:

--no-start

gdb will not be automatically started. Relevant information for starting **gdb** is printed on screen. This is somehow useful when not directly using **gdb** but some graphical front-ends, like **ddd** or **kgbd**.

--port *port*
Start the **gdb** server on the given port. If this option is not specified, a randomly chosen port will be used. If **--no-start** is specified, the port used will be printed on startup.

--with-xterm
This will run **gdb** in its own xterm instead of using the current Unix console for textual display.

In all modes, the rest of the command line, when passed, is used to identify which programs, if any, has to debugged:

program_name
This is the name of an executable to start for a debugging session. **winedbg** will actually create a process with this executable. If *programs_arguments* are also given, they will be used as arguments for creating the process to be debugged.

wpid **winedbg** will attach to the process which Windows pid is *wpid*. Use the **info proc** command within **winedbg** to list running processes and their Windows pids.

default If nothing is specified, you will enter the debugger without any run nor attached process. You'll have to do the job yourself.

COMMANDS

Default mode, and while reloading a minidump file:

Most of commands used in **winedbg** are similar to the ones from **gdb**. Please refer to the **gdb** documentations for some more details. See the *gdb differences* section later on to get a list of variations from **gdb** commands.

Misc. commands

abort Aborts the debugger.

quit Exits the debugger.

attach *N*
Attach to a Wine process (*N* is its Windows ID, numeric or hexadecimal). IDs can be obtained using the **info process** command. Note the **info process** command returns hexadecimal values

detach Detach from a Wine-process.

Help commands

help Prints some help on the commands.

help info
Prints some help on info commands

Flow control commands

cont Continue execution until next breakpoint or exception.

pass Pass the exception event up to the filter chain.

step Continue execution until next C line of code (enters function call)

next Continue execution until next C line of code (doesn't enter function call)

stepi Execute next assembly instruction (enters function call)

nexti Execute next assembly instruction (doesn't enter function call)

finish Execute until return of current function is reached.

cont, **step**, **next**, **stepi**, **nexti** can be postfixed by a number (N), meaning that the command must be executed N times before control is returned to the user.

Breakpoints, watchpoints

enable *N*
Enables (break|watch)-point *N*

disable *N*
Disables (break|watch)-point *N*

delete *N*
Deletes (break|watch)-point *N*

cond *N* Removes any existing condition to (break|watch)-point *N*

cond *N expr*
Adds condition *expr* to (break|watch)-point *N*. *expr* will be evaluated each time the (break|watch)-point is hit. If the result is a zero value, the breakpoint isn't triggered.

break * *N*
Adds a breakpoint at address *N*

break *id*
Adds a breakpoint at the address of symbol *id*

break *id N*
Adds a breakpoint at the line *N* inside symbol *id*.

break *N*
Adds a breakpoint at line *N* of current source file.

break Adds a breakpoint at current **$PC** address.

watch * *N*
Adds a watch command (on write) at address *N* (on 4 bytes).

watch *id*
Adds a watch command (on write) at the address of symbol *id*. Size depends on size of *id*.

rwatch * *N*
Adds a watch command (on read) at address *N* (on 4 bytes).

rwatch *id*
Adds a watch command (on read) at the address of symbol *id*. Size depends on size of *id*.

info break
Lists all (break|watch)-points (with their state).

You can use the symbol **EntryPoint** to stand for the entry point of the Dll.

When setting a (break|watch)-point by *id*, if the symbol cannot be found (for example, the symbol is contained in a not yet loaded module), **winedbg** will recall the name of the symbol and will try to set the breakpoint each time a new module is loaded (until it succeeds).

Stack manipulation

bt Print calling stack of current thread.

bt *N* Print calling stack of thread of ID *N*. Note: this doesn't change the position of the current frame as manipulated by the **up** & **dn** commands).

up Goes up one frame in current thread's stack

up *N* Goes up *N* frames in current thread's stack

dn Goes down one frame in current thread's stack

dn *N* Goes down *N* frames in current thread's stack

frame *N*
Sets *N* as the current frame for current thread's stack.

info locals
Prints information on local variables for current function frame.

Directory & source file manipulation

show dir
Prints the list of dirs where source files are looked for.

dir *pathname*
Adds *pathname* to the list of dirs where to look for source files

dir Deletes the list of dirs where to look for source files

symbolfile *pathname*
Loads external symbol definition file *pathname*

symbolfile *pathname N*
Loads external symbol definition file *pathname* (applying an offset of *N* to addresses)

list Lists 10 source lines forwards from current position.

list - Lists 10 source lines backwards from current position

list *N* Lists 10 source lines from line *N* in current file

list *pathname*:*N*
Lists 10 source lines from line *N* in file *pathname*

list *id* Lists 10 source lines of function *id*

list * *N* Lists 10 source lines from address *N*

You can specify the end target (to change the 10 lines value) using the ',' separator. For example:

list 123, 234
lists source lines from line 123 up to line 234 in current file

list foo.c:1,56
lists source lines from line 1 up to 56 in file foo.c

Displaying

A display is an expression that's evaluated and printed after the execution of any **winedbg** command.

display

info display
Lists the active displays

display *expr*
Adds a display for expression *expr*

display */fmt expr*
Adds a display for expression *expr*. Printing evaluated *expr* is done using the given format (see **print command** for more on formats)

del display *N*

undisplay *N*
Deletes display *N*

Disassembly

disas Disassemble from current position

disas *expr*
Disassemble from address *expr*

disas *expr*,*expr*
Disassembles code between addresses specified by the two expressions

Memory (reading, writing, typing)

x *expr* Examines memory at address *expr*

x */fmt expr*
Examines memory at address *expr* using format *fmt*

print *expr*
Prints the value of *expr* (possibly using its type)

print */fmt expr*
Prints the value of *expr* (possibly using its type)

set *var* **=** *expr*
Writes the value of *expr* in *var* variable

whatis *expr*
Prints the C type of expression *expr*

fmt is either *letter* or *count letter*, where *letter* can be:

s	an ASCII string
u	a UTF16 Unicode string
i	instructions (disassemble)
x	32-bit unsigned hexadecimal integer
d	32-bit signed decimal integer
w	16-bit unsigned hexadecimal integer
c	character (only printable 0x20-0x7f are actually printed)
b	8-bit unsigned hexadecimal integer
g	Win32 GUID

Expressions

Expressions in Wine Debugger are mostly written in a C form. However, there are a few discrepancies:

Identifiers can take a '!' in their names. This allows mainly to specify a module where to look the ID from, e.g. *USER32!CreateWindowExA*.

In a cast operation, when specifying a structure or a union, you must use the struct or union keyword (even if your program uses a typedef).

When specifying an identifier, if several symbols with this name exist, the debugger will prompt for the symbol you want to use. Pick up the one you want from its number.

Misc.

minidump *file.mdmp* saves the debugging context of the debuggee into a minidump file called *file.mdmp*.

Information on Wine internals

info class
Lists all Windows classes registered in Wine

info class *id*
Prints information on Windows class *id*

info share
Lists all the dynamic libraries loaded in the debugged program (including .so files, NE and PE DLLs)

info share *N*
Prints information on module at address *N*

info regs
Prints the value of the CPU registers

info all-regs
Prints the value of the CPU and Floating Point registers

info segment
Lists all allocated segments (i386 only)

info segment *N*
Prints information on segment *N* (i386 only)

info stack
Prints the values on top of the stack

info map
Lists all virtual mappings used by the debugged program

info map *N*
Lists all virtual mappings used by the program of Windows pid *N*

info wnd
Displays the window hierarchy starting from the desktop window

info wnd *N*
Prints information of Window of handle *N*

info process
Lists all w-processes in Wine session

info thread
Lists all w-threads in Wine session

info frame
Lists the exception frames (starting from current stack frame). You can also pass, as optional argument, a thread id (instead of current thread) to examine its exception frames.

Debug messages can be turned on and off as you are debugging using the **set** command, but only for channels initialized with the *WINEDEBUG* environment variable.

set warn + *win*
Turns on warn on *win* channel

set + *win*
Turns on warn/fixme/err/trace on *win* channel

set - *win*
Turns off warn/fixme/err/trace on *win* channel

set fixme - all
Turns off fixme class on all channels

Gdb mode:
See the **gdb** documentation for all the **gdb** commands.

However, a few Wine extensions are available, through the **monitor** command:

monitor wnd
Lists all windows in the Wine session

monitor proc
Lists all processes in the Wine session

monitor mem
Displays memory mapping of debugged process

Auto and minidump modes:
Since no user input is possible, no commands are available.

ENVIRONMENT

WINE_GDB
When used in **gdb** proxy mode, **WINE_GDB** specifies the name (and the path) of the executable to be used for **gdb**. "gdb" is used by default.

AUTHORS

The first version was written by Eric Youngdale.

See Wine developers list for the rest of contributors.

BUGS

Bugs can be reported on the **Wine bug tracker** ⟨https://bugs.winehq.org⟩.

AVAILABILITY

winedbg is part of the Wine distribution, which is available through WineHQ, the **Wine development headquarters** ⟨https://www.winehq.org/⟩.

SEE ALSO

wine(1),
Wine documentation and support ⟨https://www.winehq.org/help⟩.