NAME

winedump − A Wine DLL tool

SYNOPSIS

winedump [**-h** | **sym** *sym* | **spec** *dll* | **dump** *file*] [*mode_options*]

DESCRIPTION

winedump is a Wine tool which aims to help:
A: Reimplementing a Win32 DLL for use within Wine, or
B: Compiling a Win32 application with Winelib that uses x86 DLLs

For both tasks in order to be able to link to the Win functions some
glue code is needed. This 'glue' comes in the form of a *.spec* file.
The *.spec* file, along with some dummy code, is used to create a
Wine *.so* corresponding to the Windows DLL. The **winebuild** program
can then resolve calls made to DLL functions.

Creating a *.spec* file is a labour intensive task during which it is
easy to make a mistake. The idea of **winedump** is to automate this task
and create the majority of the support code needed for your DLL. In
addition you can have **winedump** create code to help you re-implement a
DLL, by providing tracing of calls to the DLL, and (in some cases)
automatically determining the parameters, calling conventions, and
return values of the DLL functions.

Another use for this tool is to display (dump) information about a 32bit
DLL or PE format image file. When used in this way **winedump** functions
similarly to tools such as pedump provided by many Win32 compiler
vendors.

Finally **winedump** can be also used to demangle C++ symbols.

MODES

winedump can be used in several different modes. The first argument to the program determines the mode **winedump** will run in.

-h Help mode. Basic usage help is printed.

dump To dump the contents of a file.

spec For generating .spec files and stub DLLs.

sym Symbol mode. Used to demangle C++ symbols.

OPTIONS

Mode options depend on the mode given as the first argument.

Help mode:
No options are used.
The program prints the help info and then exits.

Dump mode:

file Dumps the contents of *file*. Various file formats are supported
(PE, NE, LE, Minidumps, .lnk).

-C Turns on symbol demangling.

-f Dumps file header information.
This option dumps only the standard PE header structures,
along with the COFF sections available in the file.

-j *dir_name*
Dumps only the content of directory *dir_name*, for files
which header points to directories.

For PE files, currently the import, export, debug, resource, tls and clr directories are implemented.
For NE files, currently the export and resource directories are implemented.

-x Dumps everything.
This command prints all available information (including all available directories - see **-j** option) about the file. You may wish to pipe the output through **more/less** or into a file, since a lot of output will be produced.

-G Dumps contents of debug section if any (for now, only stabs information is supported).

Spec mode:

dll Use *dll* for input file and generate implementation code.

-I *dir* Look for prototypes in *dir* (implies **-c**). In the case of Windows DLLs, this could be either the standard include directory from your compiler, or a SDK include directory.
If you have a text document with prototypes (such as documentation) that can be used also, however you may need to delete some non-code lines to ensure that prototypes are parsed correctly.
The *dir* argument can also be a file specification (e.g. *include/**). If it contains wildcards you must quote it to prevent the shell from expanding it.
If you have no prototypes, specify */dev/null* as *dir*. **winedump** may still be able to generate some working stub code for you.

-c Generate skeleton code (requires **-I**).
This option tells **winedump** to create function stubs for each function in the DLL. As **winedump** reads each exported symbol from the source DLL, it first tries to demangle the name. If the name is a C++ symbol, the arguments, class and return value are all encoded into the symbol name. Winedump converts this information into a C function prototype. If this fails, the file(s) specified in the **-I** argument are scanned for a function prototype. If one is found it is used for the next step of the process, code generation.

-t TRACE arguments (implies **-c**).
This option produces the same code as **-c**, except that arguments are printed out when the function is called. Structs that are passed by value are printed as "struct", and functions that take variable argument lists print "...".

-f *dll* Forward calls to *dll* (implies **-t**).
This is the most complicated level of code generation. The same code is generated as **-t**, however support is added for forwarding calls to another DLL. The DLL to forward to is given as *dll*.

-D Generate documentation.
By default, **winedump** generates a standard comment at the header of each function it generates. Passing this option makes **winedump** output a full header template for standard Wine documentation, listing the parameters and return value

of the function.

-o *name*
Set the output dll name (default: **dll**).
By default, if **winedump** is run on DLL *foo*, it creates files *foo.spec*, *foo_main.c* etc, and prefixes any functions generated with *FOO_*. If **-o** *bar* is given, these will become *bar.spec*, *bar_main.c* and *BAR_* respectively.
This option is mostly useful when generating a forwarding DLL.

-C Assume __cdecl calls (default: __stdcall).
If winebuild cannot determine the calling convention, __stdcall is used by default, unless this option has been given.
Unless **-q** is given, a warning will be printed for every function that **winedump** determines the calling convention for and which does not match the assumed calling convention.

-s *num* Start prototype search after symbol *num*.

-e *num* End prototype search after symbol *num*.
By passing the **-s** or **-e** options you can have **winedump** try to generate code for only some functions in your DLL. This may be used to generate a single function, for example, if you wanted to add functionality to an existing DLL.

-S *symfile*
Search only prototype names found in *symfile*.
If you want to only generate code for a subset of exported functions from your source DLL, you can use this option to provide a text file containing the names of the symbols to extract, one per line. Only the symbols present in this file will be used in your output DLL.

-q Don't show progress (quiet).
No output is printed unless a fatal error is encountered.

-v Show lots of detail while working (verbose).
There are 3 levels of output while **winedump** is running. The default level, when neither **-q** or **-v** are given, prints the number of exported functions found in the dll, followed by the name of each function as it is processed, and a status indication of whether it was processed OK. With **-v** given, a lot of information is dumped while **winedump** works: this is intended to help debug any problems.

Sym mode:

sym Demangles C++ symbol *sym* and then exits.

FILES

function_grep.pl
Perl script used to retrieve a function prototype.

Files output in **spec** mode for *foo.dll*:
foo.spec
This is the *.spec* file.
foo_dll.h
foo_main.c
These are the source code files containing the minimum set

of code to build a stub DLL. The C file contains one function, *FOO_Init*, which does nothing (but must be present).

Makefile.in
: This is a template for **configure** to produce a makefile. It is designed for a DLL that will be inserted into the Wine source tree.

BUGS

C++ name demangling is not fully in sync with the implementation in msvcrt. It might be useful to submit your C++ name to the testsuite for msvcrt.

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

AUTHORS

Jon P. Griffiths <jon_p_griffiths at yahoo dot com>
Michael Stefaniuc <mstefani at redhat dot com>

AVAILABILITY

winedump 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⟩.