NAME

winemaker – generate a build infrastructure for compiling Windows programs on Unix

SYNOPSIS

winemaker [**--nobanner**] [**--backup** | **--nobackup**] [**--nosource-fix**]
[**--lower-none** | **--lower-all** | **--lower-uppercase**]
[**--lower-include** | **--nolower-include**] [**--mfc** | **--nomfc**]
[**--guiexe** | **--windows** | **--cuiexe** | **--console** | **--dll** | **--lib**]
[**-D***macro*[=*defn*]] [**-I***dir*] [**-P***dir*] [**-i***dll*] [**-L***dir*] [**-l***library*]
[**--nodlls**] [**--nomsvcrt**] [**--interactive**] [**--single-target** *name*]
[**--generated-files**] [**--nogenerated-files**]
[**--wine32**]
work_directory | *project_file* | *workspace_file*

DESCRIPTION

winemaker is a perl script designed to help you bootstrap the process of converting your Windows sources to Winelib programs.

In order to do this **winemaker** can perform the following operations:

- rename your source files and directories to lowercase in the event they got all uppercased during the transfer.

- perform DOS to Unix (CRLF to LF) conversions.

- scan the include statements and resource file references to replace the backslashes with forward slashes.

- during the above step **winemaker** will also perform a case insensitive search of the referenced file in the include path and rewrite the include statement with the right case if necessary.

- **winemaker** will also check other more exotic issues like *#pragma pack* usage, use of *afxres.h* in non MFC projects, and more. Whenever it encounters something out of the ordinary, it will warn you about it.

- **winemaker** can also scan a complete directory tree at once, guess what are the executables and libraries you are trying to build, match them with source files, and generate the corresponding *Makefile*.

- finally **winemaker** will generate a global *Makefile* for normal use.

- **winemaker** knows about MFC-based project and will generate customized files.

- **winemaker** can read existing project files. It supports dsp, dsw, vcproj and sln files.

OPTIONS

--nobanner
Disable the printing of the banner.

--backup
Perform a backup of all the modified source files. This is the default.

--nobackup
Do not backup modified source files.

--nosource-fix
Do no try to fix the source files (e.g. DOS to Unix conversion). This prevents complaints if the files are readonly.

--lower-all
Rename all files and directories to lowercase.

--lower-uppercase
Only rename files and directories that have an all uppercase name. So *HELLO.C* would be renamed but not *World.c*.

--lower-none
Do not rename files and directories to lower case. Note that this does not prevent the renaming of a file if its extension cannot be handled as is, e.g. ".Cxx". This is the default.

--lower-include
When the file corresponding to an include statement (or other form of file reference for resource files) cannot be found, convert that filename to lowercase. This is the default.

--nolower-include
Do not modify the include statement if the referenced file cannot be found.

--guiexe | --windows
Assume a graphical application when an executable target or a target of unknown type is found. This is the default.

--cuiexe | --console
Assume a console application when an executable target or a target of unknown type is found.

--dll Assume a dll when a target of unknown type is found, i.e. when **winemaker** is unable to determine whether it is an executable, a dll, or a static library,

--lib Assume a static library when a target of unknown type is found, i.e. when **winemaker** is unable to determine whether it is an executable, a dll, or a static library,

--mfc Specify that the targets are MFC based. In such a case **winemaker** adapts the include and library paths accordingly, and links the target with the MFC library.

--nomfc
Specify that targets are not MFC-based. This option disables use of MFC libraries even if **winemaker** encounters files *stdafx.cpp* or *stdafx.h* that would cause it to enable MFC automatically if neither **--nomfc** nor **--mfc** was specified.

-D*macro*[**=***defn*]
Add the specified macro definition to the global list of macro definitions.

-I*dir* Append the specified directory to the global include path.

-P*dir* Append the specified directory to the global dll path.

-i*dll* Add the Winelib library to the global list of Winelib libraries to import.

-L*dir* Append the specified directory to the global library path.

-l*library*
Add the specified library to the global list of libraries to link with.

--nodlls
Do not use the standard set of Winelib libraries for imports. That is, any DLL your code uses must be explicitly passed with **-i** options. The standard set of libraries is: *odbc32.dll*, *odbccp32.dll*, *ole32.dll*, *oleaut32.dll* and *winspool.drv*.

--nomsvcrt
Set some options to tell **winegcc** not to compile against msvcrt. Use this option if you have cpp-files that include *<string>*.

--interactive
Use interactive mode. In this mode **winemaker** will ask you to confirm the list of targets for each directory, and then to provide directory and target specific options.

--single-target *name*
Specify that there is only one target, called *name*.

--generated-files
Generate the *Makefile*. This is the default.

--nogenerated-files
Do not generate the *Makefile*.

--wine32
Generate a 32-bit target. This is useful on wow64 systems. Without that option the default architecture is used.

EXAMPLES

Here is a typical **winemaker** use:

$ winemaker --lower-uppercase -DSTRICT .

The above tells **winemaker** to scan the current directory and its subdirectories for source files. Whenever if finds a file or directory which name is all uppercase, it should rename it to lowercase. It should then fix all these source files for compilation with Winelib and generate *Makefile*s. The **-DSTRICT** specifies that the **STRICT** macro must be set when compiling these sources. Finally a *Makefile* will be created.

The next step would be:

$ make

If at this point you get compilation errors (which is quite likely for a reasonably sized project) then you should consult the Winelib User Guide to find tips on how to resolve them.

For an MFC-based project you would have to run the following commands instead:

$ winemaker --lower-uppercase --mfc .
$ make

For an existing project-file you would have to run the following commands:

$ winemaker myproject.dsp
$ make

TODO / BUGS

In some cases you will have to edit the *Makefile* or source files manually.

Assuming that the windows executable/library is available, we could use **winedump** to determine what kind of executable it is (graphical or console), which libraries it is linked with, and which functions it exports (for libraries). We could then restore all these settings for the corresponding Winelib target.

Furthermore **winemaker** is not very good at finding the library containing the executable: it must either be in the current directory or in the **LD_LIBRARY_PATH**.

winemaker does not support message files and the message compiler yet.

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

AUTHORS

François Gouget for CodeWeavers
Dimitrie O. Paun
André Hentschel

AVAILABILITY

winemaker 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⟩.