NAME

unifdef, **unifdefall** - remove preprocessor conditionals from code

SYNOPSIS

unifdef [**-bBcdehKkmnsStV**] [**-I***path*] [**-[i]D***sym*[*=val*]] [**-[i]U***sym*] ... [**-f** *defile*] [**-x** {*012*}] [**-M** *backext*] [**-o** *outfile*] [*infile ...*]

unifdefall [**-I***path*] *... file*

DESCRIPTION

The **unifdef** utility selectively processes conditional cpp(1) directives. It removes from a file both the directives and any additional text that they specify should be removed, while otherwise leaving the file alone.

The **unifdef** utility acts on **#if**, **#ifdef**, **#ifndef**, **#elif**, **#else**, and **#endif** lines, using macros specified in **-D** and **-U** command line options or in **-f** definitions files. A directive is processed if the macro specifications are sufficient to provide a definite value for its control expression. If the result is false, the directive and the following lines under its control are removed. If the result is true, only the directive is removed. An **#ifdef** or **#ifndef** directive is passed through unchanged if its controlling macro is not specified. Any **#if** or **#elif** control expression that has an unknown value or that **unifdef** cannot parse is passed through unchanged. By default, **unifdef** ignores **#if** and **#elif** lines with constant expressions; it can be told to process them by specifying the **-k** flag on the command line.

It understands a commonly-used subset of the expression syntax for **#if** and **#elif** lines: integer constants, integer values of macros defined on the command line, the **defined**() operator, the operators **!**, **~**, **-** (unary), *****, /, **%**, +, -, <, <=, >, >=, ==, **!=**, **&**, ^, |, **&&**, ||, and parenthesized expressions. Division by zero is treated as an unknown value. A kind of "short circuit" evaluation is used for the **&&** operator: if either operand is definitely false then the result is false, even if the value of the other operand is unknown. Similarly, if either operand of || is definitely true then the result is true.

When evaluating an expression, **unifdef** does not expand macros first. The value of a macro must be a simple number, not an expression. A limited form of indirection is allowed, where one macro’s value is the name of another.

In most cases, **unifdef** does not distinguish between object-like macros (without arguments) and function-like macros (with arguments). A function-like macro invocation can appear in **#if** and **#elif** control expressions. If the macro is not explicitly defined, or is defined with the **-D** flag on the command-line, or with **#define** in a **-f** definitions file, its arguments are ignored. If a macro is explicitly undefined on the command line with the **-U** flag, or with **#undef** in a **-f** definitions file, it may not have any arguments since this leads to a syntax error.

The **unifdef** utility understands just enough about C to know when one of the directives is inactive because it is inside a comment, or cannot be evaluated because it is split by a backslash-continued line. It spots unusually-formatted preprocessor directives and passes them through unchanged when the layout is too odd for it to handle. (See the *BUGS* section below.)

A script called **unifdefall** can be used to remove all conditional cpp(1) directives from a file. It uses **unifdef -s** and **cpp -dM** to get lists of all the controlling macros and their definitions (or lack thereof), then invokes **unifdef** with appropriate arguments to process the file.

OPTIONS

-D*sym*=*val*

Specify that a macro is defined to a given value.

-D*sym*

Specify that a macro is defined to the value 1.

-U*sym*

Specify that a macro is undefined.

If the same macro appears in more than one argument, the last occurrence dominates.

-iD*sym*[=*val*]
-iU*sym*

C strings, comments, and line continuations are ignored within **#ifdef** and **#ifndef** blocks controlled by macros specified with these options.

-f *defile*

The file *defile* contains **#define** and **#undef** preprocessor directives, which have the same effect as the corresponding **-D** and **-U** command-line arguments. You can have multiple **-f** arguments and mix them with **-D** and **-U** arguments; later options override earlier ones.

Each directive must be on a single line. Object-like macro definitions (without arguments) are set to the given value. Function-like macro definitions (with arguments) are treated as if they are set to 1.

Warning: string literals and character constants are not parsed correctly in **-f** files.

-b Replace removed lines with blank lines instead of deleting them. Mutually exclusive with the **-B** option.

-B Compress blank lines around a deleted section. Mutually exclusive with the **-b** option.

-c Complement, i.e., lines that would have been removed or blanked are retained and vice versa.

-d Turn on printing of debugging messages.

-e By default, **unifdef** will report an error if it needs to remove a preprocessor directive that spans more than one line, for example, if it has a multi-line comment hanging off its right hand end. The **-e** flag makes it ignore the line instead.

-h Print help.

-I*path* Specifies to **unifdefall** an additional place to look for **#include** files. This option is ignored by **unifdef** for compatibility with cpp(1) and to simplify the implementation of **unifdefall**.

-K Always treat the result of **&&** and **||** operators as unknown if either operand is unknown, instead of short-circuiting when unknown operands can't affect the result. This option is for compatibility with older versions of **unifdef**.

-k Process **#if** and **#elif** lines with constant expressions. By default, sections controlled by such lines are passed through unchanged because they typically start "#if 0" and are used as a kind of comment to sketch out future or past development. It would be rude to strip them out, just as it would be for normal comments.

-m Modify one or more input files in place. If an input file is not modified, the original is preserved instead of being overwritten with an identical copy.

-M *backext*
Modify input files in place, and keep backups of the original files by appending the *backext* to the input filenames. A zero length *backext* behaves the same as the **-m** option.

-n Add #line directives to the output following any deleted lines, so that errors produced when compiling the output file correspond to line numbers in the input file.

-o *outfile*
Write output to the file *outfile* instead of the standard output when processing a single file.

-s Instead of processing an input file as usual, this option causes **unifdef** to produce a list of macros that are used in preprocessor directive controlling expressions.

-S Like the **-s** option, but the nesting depth of each macro is also printed. This is useful for working out the number of possible combinations of interdependent defined/undefined macros.

-t Disables parsing for C strings, comments, and line continuations, which is useful for plain text. This is a blanket version of the **-iD** and **-iU** flags.

-V Print version details.

-x {*012*}
Set exit status mode to zero, one, or two. See the *EXIT STATUS* section below for details.

The **unifdef** utility takes its input from *stdin* if there are no *file* arguments. You must use the **-m** or **-M** options if there are multiple input files. You can specify input from stdin or output to stdout with '-'.

The **unifdef** utility works nicely with the **-D***sym* option of diff(1).

EXIT STATUS

In normal usage the **unifdef** utility's exit status depends on the mode set using the **-x** option.

If the exit mode is zero (the default) then **unifdef** exits with status 0 if the output is an exact copy of the input, or with status 1 if the output differs.

If the exit mode is one, **unifdef** exits with status 1 if the output is unmodified or 0 if it differs.

If the exit mode is two, **unifdef** exits with status zero in both cases.

In all exit modes, **unifdef** exits with status 2 if there is an error.

The exit status is 0 if the **-h** or **-V** command line options are given.

DIAGNOSTICS

EOF in comment

Inappropriate **#elif**, **#else** or **#endif**

Missing macro name in #define or #undef

Obfuscated preprocessor control line

Premature EOF (with the line number of the most recent unterminated **#if**)

Too many levels of nesting

Unrecognized preprocessor directive

Unterminated char or string literal

SEE ALSO

cpp(1), diff(1)

The unifdef home page is *http://dotat.at/prog/unifdef*

HISTORY

The **unifdef** command appeared in 2.9BSD. ANSI C support was added in FreeBSD 4.7.

AUTHORS

The original implementation was written by Dave Yost <*Dave@Yost.com*>. Tony Finch <*dot@dotat.at*> rewrote it to support ANSI C.

BUGS

- Expression evaluation is very limited.
- Character constants are not evaluated. String literals and character constants in **-f** definition files are ignored rather than parsed as part of a macro's replacement tokens.
- Only the basic form of C++ raw string literals is recognized, like R"(string)" without delimiters as in R"delimiter(string)delimiter".
- Source files are processed one line at a time, so preprocessor directives split across more than one physical line (because of comments or backslash-newline) cannot be handled in every situation.
- Trigraphs are not recognized.
- There is no support for macros with different definitions at different points in the source file.
- The text-mode and ignore functionality does not correspond to modern cpp(1) behaviour.

Please send bug reports by email to <*dot@dotat.at*>.