hexproc(1) Manual Page

Name

hexproc - hexadecimal preprocessor for building binary files or hexdumps

Synopsis

`hexproc [OPTION…] [FILE]`

Options

-v

Prints the program version and exits

-V

Prints advanced program configuration and exits

-h

Prints a brief help message and exits

-b

Outputs binary data instead of hexadecimal

-B

Forces writing binary data even if output is a TTY

-c

Enable colored output - bytes resulting from a single formatter or string will be marked with the same color; not compatible with **-b** or **-B**

-C

Forces colored output even if the output is not a TTY

-d

Enter debug mode

Description

Hexproc is a tool for building hex files. The input file (or `stdin`, if no file given) is processed and then written to `stdout`.

Hexproc is meant to be used as part of a pipeline and therefore does not implement many features which are provided by other processing tools.

It is recommended to use a text preprocessor (such as **cpp** or **m4**) before feeding the input to **hexproc** as **hexproc** does not provide macros or file inclusion.

Syntax

Hexproc input consists of the following tokens:

strings

Quoted ASCII text is converted to bytes and replaced with hexadecimal octets. For example, the result of **"Hello"** is **48 65 6c 6c 6f**

octets

Two hexadecimal digits will appear on the output as-is.

comments

End-of-line comments are created using # or //; Note that comments in form of #*NUMBER* "*filename*" are recognized as line markers, similar to how the C preprocessor uses them. This way, you can get correct debug information after using **#include**. Block comments start with /* and end with */. They cannot be nested.

labels

The syntax *NAME* **:** will assign the current byte offset to the specified name. Labels may freely appear between tokens.

lazy assignment

The syntax *NAME* = *EXPRESSION* will map the variable name to the given expression without evaluating it. Whenever the name is referenced, the expression will be evaluated. The expression must not be recursive. If you want to add more tokens after an assignment, terminate the expression with ";".

immediate assignment

The syntax *NAME* := *EXPRESSION* will evaluate the expression and assign the result to the variable. The expression may reference the variable it's being assigned to. If you want to add more tokens after an assignment, terminate the expression with ";".

formatters

The syntax [*attr1*, *attr2*, ...](*EXPRESSION*) is called a formatter. Parentheses around *EXPRESSION* can be omitted if the expression is a single number or variable name. The commas are optional. Each *attr* can be one of the following:

- an integer between 0 and 8 inclusive, denoting the number of bytes to use
- the string **BE** or **LE**, which changes the representation of the value to big or little endian, respectively
- a type name, such as **int64** or **float**, which controls the size and representation of the value. (See section **Type Names** for more information).

Leading and trailing whitespace is ignored.

Hexproc maps lines one-to-one so that line numbers are preserved.

Expression Syntax

Expressions are not standalone constructs, they are used in assignments and formatters. Expressions support parentheses and the following binary operators: +, **-**, ***, /, ^ (exponentiation), **%** (remainder), **&** (binary and), | (bitwise or) and ~ (bitwise xor). Unary minus and binary not are also supported. You can use decimal, hexadecimal (prefix with **0x**) or octal (prefix with **0**) number literals. The number literal format is specified by the compiler used to build hexproc. You can also refer to variables by their names. Operator precedence works just like in C.

Special Variables

A few variables have a special role in hexproc. These variable names start with the prefix "**hexproc.**"

- **hexproc.endian** - sets the default endianness of formatters
- **hexproc.major** - version release number of hexproc ("**1**" in "**v1.2.3**")
- **hexproc.minor** - minor version number of hexproc ("**2**" in "**v1.2.3**")
- **hexproc.patch** - patch number of hexproc ("**3**" in "**v1.2.3**")

Type Names

The following identifiers may be used as type names:

- **int64**, **int32**, **int16**, **int8** - fixed size types
- **long**, **int**, **short**, **byte** - synonymous with fixed size types
- **ieee754_single**, **ieee754_double** - IEEE 754 single and double precision floating point types
- **float**, **double** - synonymous with above floating point types

---

Debugger

Hexproc comes with a built-in debugger, which you can activate using the **-d** option. The debugger supports the following commands:

break *NUMBER*, b *NUMBER*

Creates a breakpoint on given line number. When execution reaches a line with a breakpoint, it will enter the debugger before evaluating the line

resume, r

Suspends the debugger and resumes execution.

step, s

Executes next line and enters debugger again.

vars, v, list, l

Displays the current state of all variables

help, h, ?

Shows help on how to use the debugger

eval *EXPR*, e *EXPR*, = *EXPR*

Evaluate an expression

See Also

xxd(1), cpp(1)

Version 0.3.5
Last updated 2021-01-05 16:21:21 +0200