ECLAIR

This page was automatically generated and should not be edited.

The information on this page was provided by outside contributors and has not been verified by SEI CERT.

The table below can be re-ordered, by clicking column headers.

Tool Version: 1.2

Checker	Guideline
CC2.API08	API08-C. Avoid parameter names in a function prototype
CC2.ARR02	ARR02-C. Explicitly specify array bounds, even if implicitly defined by an initializer
CC2.DCL00	DCL00-C. Const-qualify immutable objects
CC2.DCL01	DCL01-C. Do not reuse variable names in subscopes
CC2.DCL02	DCL02-C. Use visually distinct identifiers
CC2.DCL03	DCL03-C. Use a static assertion to test the value of a constant expression
CC2.DCL04	DCL04-C. Do not declare more than one variable per declaration
CC2.DCL06	DCL06-C. Use meaningful symbolic constants to represent literal values
CC2.DCL07	DCL07-C. Include the appropriate type information in function declarators
CC2.DCL11	DCL11-C. Understand the type issues associated with variadic functions
CC2.DCL13	DCL13-C. Declare function parameters that are pointers to values not changed by the function as const
CC2.DCL15	DCL15-C. Declare file-scope objects or functions that do not need external linkage as static
CC2.DCL16	DCL16-C. Use "L," not "l," to indicate a long value
CC2.DCL19	DCL19-C. Minimize the scope of variables and functions
CC2.DCL31	DCL31-C. Declare identifiers before using them
CC2.DCL36	DCL36-C. Do not declare an identifier with conflicting linkage classifications
CC2.DCL37	DCL37-C. Do not declare or define a reserved identifier
CC2.ERR01	ERR01-C. Use ferror() rather than errno to check for FILE stream errors
CC2.EXP00	EXP00-C. Use parentheses for precedence of operation
CC2.EXP05	EXP05-C. Do not cast away a const qualification
CC2.EXP06	EXP44-C. Do not rely on side effects in operands to sizeof, _Alignof, or _Generic
CC2.EXP09	EXP09-C. Use sizeof to determine the size of a type or variable
CC2.EXP12	EXP12-C. Do not ignore values returned by functions
CC2.EXP13	EXP13-C. Treat relational and equality operators as if they were nonassociative
CC2.EXP14	EXP14-C. Beware of integer promotion when performing bitwise operations on integer types smaller than int
CC2.EXP18	EXP45-C. Do not perform assignments in selection statements
CC2.EXP21	EXP45-C. Do not perform assignments in selection statements
CC2.EXP30	EXP30-C. Do not depend on the order of evaluation for side effects
CC2.EXP31	PRE31-C. Avoid side effects in arguments to unsafe macros
CC2.EXP36	EXP36-C. Do not cast pointers into more strictly aligned pointer types
CC2.EXP37	EXP37-C. Call functions with the correct number and type of arguments
CC2.FIO34	FIO34-C. Distinguish between characters read from a file and EOF or WEOF
CC2.FLP00	FLP00-C. Understand the limitations of floating-point numbers
CC2.FLP30	FLP30-C. Do not use floating-point variables as loop counters

CC2.INT02	INT02-C. Understand integer conversion rules
CC2.INT07	INT07-C. Use only explicitly signed or unsigned char type for numeric values
CC2.INT09	INT09-C. Ensure enumeration constants map to unique values
CC2.INT12	INT12-C. Do not make assumptions about the type of a plain int bit-field when used in an expression
CC2.INT13	INT13-C. Use bitwise operators only on unsigned operands
CC2.INT34	INT34-C. Do not shift an expression by a negative number of bits or by greater than or equal to the number of bits that exist in the operand
CC2. MEM02	MEM02-C. Immediately cast the result of a memory allocation function call into a pointer to the allocated type
CC2. MSC04	MSC04-C. Use comments consistently and in a readable fashion
CC2. MSC05	MSC05-C. Do not manipulate time_t typed values directly
CC2. MSC12	MSC12-C. Detect and remove code that has no effect or is never executed
CC2. MSC17	MSC17-C. Finish every set of statements associated with a case label with a break statement
CC2. MSC20	MSC20-C. Do not use a switch statement to transfer control into a complex block
CC2. MSC30	MSC30-C. Do not use the rand() function for generating pseudorandom numbers
CC2. MSC34	MSC24-C. Do not use deprecated or obsolescent functions
CC2.PRE00	PRE00-C. Prefer inline or static functions to function-like macros
CC2.PRE01	PRE01-C. Use parentheses within macros around parameter names
CC2.PRE02	PRE02-C. Macro replacement lists should be parenthesized
CC2.PRE03	PRE03-C. Prefer typedefs to defines for encoding non-pointer types
CC2.PRE04	PRE04-C. Do not reuse a standard header file name
CC2.PRE06	PRE06-C. Enclose header files in an include guard
CC2.PRE07	PRE07-C. Avoid using repeated question marks
CC2.PRE08	PRE08-C. Guarantee that header file names are unique
CC2.PRE12	PRE12-C. Do not define unsafe macros
CC2.PRE31	PRE31-C. Avoid side effects in arguments to unsafe macros
CC2.PRE32	PRE32-C. Do not use preprocessor directives in invocations of function-like macros
CC2.STR04	STR04-C. Use plain char for characters in the basic character set
CC2.STR05	STR05-C. Use pointers to const when referring to string literals
CC2.STR10	STR10-C. Do not concatenate different type of string literals
CC2.STR34	STR34-C. Cast characters to unsigned char before converting to larger integer sizes
CC2.STR36	STR11-C. Do not specify the bound of a character array initialized with a string literal
CC2.STR37	STR37-C. Arguments to character-handling functions must be representable as an unsigned char