CERT C

Perforce QAC for C 2026.1

CERT C 2016 Edition plus website 9 July 2025

Rule Enforcement Summary

		Total
a	Total Number of Rules	103
b	Total Number of 'Not Statically Enforceable' Rules (Assisted/Unassisted)	0
c	Total Number of Enforceable Rules (a-b)	103
d	Total Number of Enforced Rules	103
e	Total Number of Unenforced Rules (c-d)	0
f	Enforced Rules Percentage (d/c)	100%
g	Unenforced Rules Percentage (e/c)	0%

Id	Description	Level	Enforced
Preprocessor (PRE)			
PRE30-C	Do not create a universal character name through concatenation	L3	Yes
PRE31-C	Avoid side effects in arguments to unsafe macros	L2	Yes
PRE32-C	Do not use preprocessor directives in invocations of function-like macros	L3	Yes
Declarations and Initialization (DCL)			
DCL30-C	Declare objects with appropriate storage durations	L2	Yes
DCL31-C	Declare identifiers before using them	L3	Yes
DCL36-C	Do not declare an identifier with conflicting linkage classifications	L2	Yes
DCL37-C	Do not declare or define a reserved identifier	L3	Yes
DCL38-C	Use the correct syntax when declaring a flexible array member	L3	Yes

Id	Description	Level	Enforced
DCL39-C	Avoid information leakage when passing a structure across a trust boundary	L3	Yes
DCL40-C	Do not create incompatible declarations of the same function or object	L3	Yes
DCL41-C	Do not declare variables inside a switch statement before the first case label	L2	Yes
Expressions (EXP)			
EXP30-C	Do not depend on the order of evaluation for side effects	L2	Yes
EXP32-C	Do not access a volatile object through a nonvolatile reference	L2	Yes
EXP33-C	Do not read uninitialized memory	L1	Yes
EXP34-C	Do not dereference null pointers	L1	Yes
EXP35-C	Do not modify objects with temporary lifetime	L2	Yes
EXP36-C	Do not cast pointers into more strictly aligned pointer types	L3	Yes
EXP37-C	Call functions with the correct number and type of arguments	L3	Yes
EXP39-C	Do not access a variable through a pointer of an incompatible type	L3	Yes
EXP40-C	Do not modify constant objects	L3	Yes
EXP42-C	Do not compare padding data	L1	Yes
EXP43-C	Avoid undefined behavior when using restrict-qualified pointers	L3	Yes
EXP44-C	Do not rely on side effects in operands to sizeof, _Alignof, or _Generic	L3	Yes
EXP45-C	Do not perform assignments in selection statements	L2	Yes
EXP46-C	Do not use a bitwise operator with a Boolean-like operand	L2	Yes
EXP47-C	Do not call va_arg with an argument of the incorrect type	L2	Yes
Integers (INT)			
INT30-C	Ensure that unsigned integer operations do not wrap	L2	Yes
INT31-C	Ensure that integer conversions do not result in lost or misinterpreted data	L1	Yes
INT32-C	Ensure that operations on signed integers do not result in overflow	L1	Yes

Id	Description	Level	Enforced
INT33-C	Ensure that division and remainder operations do not result in divide-by-zero errors	L2	Yes
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	L3	Yes
INT35-C	Use correct integer precisions	L3	Yes
INT36-C	Converting a pointer to integer or integer to pointer	L3	Yes
Floating Point (FLP)			
FLP30-C	Do not use floating-point variables as loop counters	L2	Yes
FLP32-C	Prevent or detect domain and range errors in math functions	L2	Yes
FLP34-C	Ensure that floating-point conversions are within range of the new type	L3	Yes
FLP36-C	Preserve precision when converting integral values to floating-point type	L3	Yes
FLP37-C	Do not use object representations to compare floating-point values	L3	Yes
Arrays (ARR)			
ARR30-C	Do not form or use out-of-bounds pointers or array subscripts	L2	Yes
ARR32-C	Ensure size arguments for variable length arrays are in a valid range	L2	Yes
ARR36-C	Do not subtract or compare two pointers that do not refer to the same array	L2	Yes
ARR37-C	Do not add or subtract an integer to a pointer to a non-array object	L2	Yes
ARR38-C	Guarantee that library functions do not form invalid pointers	L1	Yes
ARR39-C	Do not add or subtract a scaled integer to a pointer	L2	Yes
Characters and Strings (STR)			
STR30-C	Do not attempt to modify string literals	L2	Yes
STR31-C	Guarantee that storage for strings has sufficient space for character data and the null terminator	L2	Yes
STR32-C	Do not pass a non-null-terminated character sequence to a library function that expects a string	L1	Yes

Id	Description	Level	Enforced
STR34-C	Cast characters to unsigned char before converting to larger integer sizes	L2	Yes
STR37-C	Arguments to character-handling functions must be representable as an unsigned char	L3	Yes
STR38-C	Do not confuse narrow and wide character strings and functions	L1	Yes
Memory Management (MEM)			
MEM30-C	Do not access freed memory	L2	Yes
MEM31-C	Free dynamically allocated memory when no longer needed	L3	Yes
MEM33-C	Allocate and copy structures containing a flexible array member dynamically	L3	Yes
MEM34-C	Only free memory allocated dynamically	L2	Yes
MEM35-C	Allocate sufficient memory for an object	L2	Yes
MEM36-C	Do not modify the alignment of objects by calling realloc()	L3	Yes
Input Output (FIO)			
FIO30-C	Exclude user input from format strings	L1	Yes
FIO32-C	Do not perform operations on devices that are only appropriate for files	L3	Yes
FIO34-C	Distinguish between characters read from a file and EOF or WEOF	L1	Yes
FIO37-C	Do not assume that fgets() or fgetws() returns a nonempty string when successful	L1	Yes
FIO38-C	Do not copy a FILE object	L3	Yes
FIO39-C	Do not alternately input and output from a stream without an intervening flush or positioning call	L2	Yes
FIO40-C	Reset strings on fgets() or fgetws() failure	L2	Yes
FIO41-C	Do not call getc(), putc(), getwc(), or putwc() with a stream argument that has side effects	L3	Yes
FIO42-C	Close files when they are no longer needed	L3	Yes
FIO44-C	Only use values for fsetpos() that are returned from fgetpos()	L3	Yes
FIO45-C	Avoid TOCTOU race conditions while accessing files	L2	Yes

Id	Description	Level	Enforced
FIO46-C	Do not access a closed file	L3	Yes
FIO47-C	Use valid format strings	L2	Yes
Environment (ENV)			
ENV30-C	Do not modify the object referenced by the return value of certain functions	L3	Yes
ENV31-C	Do not rely on an environment pointer following an operation that may invalidate it	L3	Yes
ENV32-C	All exit handlers must return normally	L1	Yes
ENV33-C	Do not call system()	L1	Yes
ENV34-C	Do not store pointers returned by certain functions	L3	Yes
Signals (SIG)			
SIG30-C	Call only asynchronous-safe functions within signal handlers	L1	Yes
SIG31-C	Do not access shared objects in signal handlers	L1	Yes
SIG34-C	Do not call signal() from within interruptible signal handlers	L3	Yes
SIG35-C	Do not return from a computational exception signal handler	L3	Yes
Error Handling (ERR)			
ERR30-C	Set errno to zero before calling a library function known to set errno, and check errno only after the function returns a value indicating failure	L1	Yes
ERR32-C	Do not rely on indeterminate values of errno	L3	Yes
ERR33-C	Detect and handle standard library errors	L1	Yes
ERR34-C	Detect errors when converting a string to a number	L2	Yes
Concurrency (CON)			
CON30-C	Clean up thread-specific storage	L3	Yes
CON31-C	Do not destroy a mutex while it is locked	L3	Yes
CON32-C	Prevent data races when accessing bit-fields from multiple threads	L3	Yes
CON33-C	Avoid race conditions when using library functions	L3	Yes

Id	Description	Level	Enforced
CON34-C	Declare objects shared between threads with appropriate storage durations	L3	Yes
CON35-C	Avoid deadlock by locking in a predefined order	L3	Yes
CON36-C	Wrap functions that can spuriously wake up in a loop	L3	Yes
CON37-C	Do not call signal() in a multithreaded program	L3	Yes
CON38-C	Preserve thread safety and liveness when using condition variables	L3	Yes
CON39-C	Do not join or detach a thread that was previously joined or detached	L3	Yes
CON40-C	Do not refer to an atomic variable twice in an expression	L2	Yes
CON41-C	Wrap functions that can fail spuriously in a loop	L3	Yes
CON43-C	Do not allow data races in multithreaded code	L3	Yes
Miscellaneous (MSC)			
MSC30-C	Do not use the rand() function for generating pseudorandom numbers	L3	Yes
MSC32-C	Properly seed pseudorandom number generators	L1	Yes
MSC33-C	Do not pass invalid data to the asctime() function	L2	Yes
MSC37-C	Ensure that control never reaches the end of a non-void function	L2	Yes
MSC38-C	Do not treat a predefined identifier as an object if it might only be implemented as a macro	L3	Yes
MSC39-C	Do not call va_arg() on a va_list that has an indeterminate value	L3	Yes
MSC40-C	Do not violate constraints	L3	Yes
MSC41-C	Never hard code sensitive information	L2	Yes

POSIX Enforcement

Id	Description	Level	Enforced
POSIX (POS)			
POS02-C	Follow the principle of least privilege	L2	Yes

Id	Description	Level	Enforced
POS30-C	Use the readlink() function properly	L1	Yes
POS34-C	Do not call putenv() with a pointer to an automatic variable as the argument	L2	Yes
POS35-C	Avoid race conditions while checking for the existence of a symbolic link	L2	Yes
POS36-C	Observe correct revocation order while relinquishing privileges	L1	Yes
POS37-C	Ensure that privilege relinquishment is successful	L1	Yes
POS38-C	Beware of race conditions when using fork and file descriptors	L3	Yes
POS39-C	Use the correct byte ordering when transferring data between systems	L1	Yes
POS44-C	Do not use signals to terminate threads	L3	Yes
POS47-C	Do not use threads that can be canceled asynchronously	L3	Yes
POS48-C	Do not unlock or destroy another POSIX thread's mutex	L3	Yes
POS49-C	When data must be accessed by multiple threads, provide a mutex and guarantee no adjacent data is also accessed	L3	Yes
POS50-C	Declare objects shared between POSIX threads with appropriate storage durations	L3	Yes
POS51-C	Avoid deadlock with POSIX threads by locking in predefined order	L3	Yes
POS52-C	Do not perform operations that can block while holding a POSIX lock	L3	Yes
POS53-C	Do not use more than one mutex for concurrent waiting operations on a condition variable	L2	Yes
POS54-C	Detect and handle POSIX library errors	L1	Yes

Recommendation Enforcement

Id	Description	Level	Enforced
Preprocessor (PRE)			
PRE00-C	Prefer inline or static functions to function-like macros	L3	Yes

Id	Description	Level	Enforced
PRE01-C	Use parentheses within macros around parameter names	L1	Yes
PRE02-C	Macro replacement lists should be parenthesized	L1	Yes
PRE03-C	Prefer typedefs to defines for encoding non-pointer types	L3	Yes
PRE04-C	Do not reuse a standard header file name	L3	Yes
PRE05-C	Understand macro replacement when concatenating tokens or performing stringification	L3	Yes
PRE06-C	Enclose header files in an inclusion guard	L3	Yes
PRE07-C	Avoid using repeated question marks	L3	Yes
PRE08-C	Guarantee that header file names are unique	L3	Yes
PRE09-C	Do not replace secure functions with deprecated or obsolescent functions	L1	Yes
PRE10-C	Wrap multi-statement macros in a do-while loop	L1	Yes
PRE11-C	Do not conclude macro definitions with a semicolon	L2	Yes
PRE12-C	Do not define unsafe macros	L3	Yes
PRE13-C	Use the Standard predefined macros to test for versions and features.	L3	Yes
Declarations and Initialization (DCL)			
DCL00-C	Const-qualify immutable objects	L3	Yes
DCL01-C	Do not reuse variable names in subscopes	L3	Yes
DCL05-C	Use typedefs of non-pointer types only	L3	Yes
DCL06-C	Use meaningful symbolic constants to represent literal values	L3	Yes
DCL07-C	Include the appropriate type information in function declarators	L3	Yes
DCL10-C	Maintain the contract between the writer and caller of variadic functions	L2	Yes
DCL11-C	Understand the type issues associated with variadic functions	L1	Yes
DCL13-C	Declare function parameters that are pointers to values not changed by the function as const	L3	Yes

Id	Description	Level	Enforced
DCL15-C	Declare file-scope objects or functions that do not need external linkage as static	L3	Yes
DCL16-C	Use 'L', not 'l', to indicate a long value	L3	Yes
DCL18-C	Do not begin integer constants with 0 when specifying a decimal value	L3	Yes
DCL19-C	Minimize the scope of variables and functions	L3	Yes
DCL20-C	Explicitly specify void when a function accepts no arguments	L1	Yes
DCL21-C	Understand the storage of compound literals	L3	Yes
DCL23-C	Guarantee that mutually visible identifiers are unique	L2	Yes
Expressions (EXP)			
EXP00-C	Use parentheses for precedence of operation	L2	Yes
EXP02-C	Be aware of the short-circuit behavior of the logical AND and OR operators	L3	Yes
EXP03-C	Do not assume the size of a structure is the sum of the sizes of its members	L3	Yes
EXP05-C	Do not cast away a const qualification	L3	Yes
EXP07-C	Do not diminish the benefits of constants by assuming their values in expressions	L3	Yes
EXP08-C	Ensure pointer arithmetic is used correctly	L2	Yes
EXP09-C	Use sizeof to determine the size of a type or variable	L2	Yes
EXP10-C	Do not depend on the order of evaluation of subexpressions or the order in which side effects take place	L2	Yes
EXP11-C	Do not make assumptions regarding the layout of structures with bit-fields	L3	Yes
EXP12-C	Do not ignore values returned by functions	L3	Yes
EXP13-C	Treat relational and equality operators as if they were nonassociative	L3	Yes
EXP15-C	Do not place a semicolon on the same line as an if, for, or while statement	L1	Yes
EXP16-C	Do not compare function pointers to constant values	L2	Yes

Id	Description	Level	Enforced
EXP19-C	Use braces for the body of an if, for, or while statement	L1	Yes
EXP20-C	Perform explicit tests to determine success, true and false, and equality	L1	Yes
Integers (INT)			
INT02-C	Understand integer conversion rules	L3	Yes
INT04-C	Enforce limits on integer values originating from tainted sources	L1	Yes
INT05-C	Do not use input functions to convert character data if they cannot handle all possible inputs	L2	Yes
INT07-C	Use only explicitly signed or unsigned char type for numeric values	L1	Yes
INT08-C	Verify that all integer values are in range	L3	Yes
INT09-C	Ensure enumeration constants map to unique values	L3	Yes
INT10-C	Do not assume a positive remainder when using the % operator	L3	Yes
INT12-C	Do not make assumptions about the type of a plain int bit-field when used in an expression	L3	Yes
INT13-C	Use bitwise operators only on unsigned operands	L2	Yes
INT16-C	Do not make assumptions about representation of signed integers	L3	Yes
INT17-C	Define integer constants in an implementation-independent manner	L3	Yes
INT18-C	Evaluate integer expressions in a larger size before comparing or assigning to that size	L1	Yes
Floating Point (FLP)			
FLP00-C	Understand the limitations of floating-point numbers	L3	Yes
FLP02-C	Avoid using floating-point numbers when precise computation is needed	L3	Yes
FLP06-C	Convert integers to floating point for floating point operations	L3	Yes
Arrays (ARR)			
ARR00-C	Understand how arrays work	L2	Yes
ARR01-C	Do not apply the sizeof operator to a pointer when taking the size of an array	L1	Yes

Id	Description	Level	Enforced
ARR02-C	Explicitly specify array bounds, even if implicitly defined by an initializer	L2	Yes
Characters and Strings (STR)			
STR00-C	Represent characters using an appropriate type	L3	Yes
STR02-C	Sanitize data passed to complex subsystems	L2	Yes
STR03-C	Do not inadvertently truncate a string	L3	Yes
STR04-C	Use plain char for characters in the basic character set	L3	Yes
STR05-C	Use pointers to const when referring to string literals	L3	Yes
STR06-C	Do not assume that strtok() leaves the parse string unchanged	L2	Yes
STR07-C	Use the bounds-checking interfaces for string manipulation	L1	Yes
STR09-C	Don't assume numeric values for expressions with type plain character	L3	Yes
STR10-C	Do not concatenate different type of string literals	L3	Yes
STR11-C	Do not specify the bound of a character array initialized with a string literal	L2	Yes
Memory Management (MEM)			
MEM00-C	Allocate and free memory in the same module, at the same level of abstraction	L1	Yes
MEM01-C	Store a new value in pointers immediately after free()	L2	Yes
MEM02-C	Immediately cast the result of a memory allocation function call into a pointer to the allocated type	L3	Yes
MEM03-C	Clear sensitive information stored in reusable resources	L3	Yes
MEM04-C	Beware of zero-length allocations	L2	Yes
MEM05-C	Avoid large stack allocations	L1	Yes
MEM07-C	Ensure that the arguments to calloc(), when multiplied, do not wrap	L2	Yes
Input Output (FIO)			
FIO01-C	Be careful using functions that use file names for identification	L1	Yes

Id	Description	Level	Enforced
FIO02-C	Canonicalize path names originating from tainted sources	L3	Yes
FIO03-C	Do not make assumptions about fopen() and file creation	L3	Yes
FIO06-C	Create files with appropriate access permissions	L3	Yes
FIO08-C	Take care when calling remove() on an open file	L3	Yes
FIO10-C	Take care when using the rename() function	L3	Yes
FIO20-C	Avoid unintentional truncation when using fgets() or fgetws()	L1	Yes
FIO21-C	Do not create temporary files in shared directories	L3	Yes
Environment (ENV)			
ENV01-C	Do not make assumptions about the size of an environment variable	L2	Yes
ENV03-C	Sanitize the environment when invoking external programs	L2	Yes
Signals (SIG)			
SIG00-C	Mask signals handled by noninterruptible signal handlers	L3	Yes
SIG01-C	Understand implementation-specific details regarding signal handler persistence	L3	Yes
SIG02-C	Avoid using signals to implement normal functionality	L2	Yes
Error Handling (ERR)			
ERR07-C	Prefer functions that support error checking over equivalent functions that don't	L1	Yes
Concurrency (CON)			
CON02-C	Do not use volatile as a synchronization primitive	L3	Yes
CON03-C	Ensure visibility when accessing shared variables	L3	Yes
CON07-C	Ensure that compound operations on shared variables are atomic	L2	Yes
Miscellaneous (MSC)			
MSC00-C	Compile cleanly at high warning levels	L3	Yes
MSC01-C	Strive for logical completeness	L3	Yes
MSC04-C	Use comments consistently and in a readable fashion	L3	Yes

Id	Description	Level	Enforced
MSC09-C	Character encoding: Use subset of ASCII for safety	L3	Yes
MSC12-C	Detect and remove code that has no effect or is never executed	L3	Yes
MSC13-C	Detect and remove unused values	L3	Yes
MSC14-C	Do not introduce unnecessary platform dependencies	L3	Yes
MSC15-C	Do not depend on undefined behavior	L2	Yes
MSC17-C	Finish every set of statements associated with a case label with a break statement	L1	Yes
MSC18-C	Be careful while handling sensitive data, such as passwords, in program code	L3	Yes
MSC20-C	Do not use a switch statement to transfer control into a complex block	L2	Yes
MSC23-C	Beware of vendor-specific library and language differences	L3	Yes
MSC24-C	Do not use deprecated or obsolescent functions	L1	Yes