BB. Definitions

abnormal end [ISO/IEC/IEEE 24765:2010]
Termination of a process prior to completion.

abnormal program termination See *abnormal end*.

analyzer [ISO/IEC TS 17961:2013]
Mechanism that diagnoses coding flaws in software programs.

> NOTE
> Analyzers may include static analysis tools, tools within a compiler suite, or tools in other contexts.

async-signal-safe function [ISO/IEC 9945:2008]
A function that may be invoked, without restriction, from signal-catching functions. No function (defined in ISO/IEC 9945) is async-signal-safe unless explicitly described as such. See *asynchronous-safe*.

asynchronous-safe function [GNU Pth]
A function is asynchronous-safe, or asynchronous-signal safe, if it can be called safely and without side effects from within a signal handler context. That is, it must be able to be interrupted at any point to run linearly out of sequence without causing an inconsistent state. It must also function properly when global data might itself be in an inconsistent state. Some asynchronous-safe operations are listed here:

- Call the `signal()` function to reinstall a signal handler.
- Unconditionally modify a `volatile sig_atomic_t` variable (as modification to this type is atomic).
- Call the `_Exit()` function to immediately terminate program execution.
- Invoke an asynchronous-safe function, as specified by the implementation.

Few functions are portably asynchronous-safe.

availability [IEEE Std 610.12 1990]
The degree to which a system or component is operational and accessible when required for use. Often expressed as a probability.

condition predicate
An expression constructed from the variables of a function that must be true for a thread to be allowed to continue execution.

conforming [ISO/IEC 9899:2011]
Conforming programs may depend on nonportable features of a conforming implementation.

critical sections
Code that accesses shared data and would otherwise be protected from data races.

dangling pointer
A pointer to deallocated memory.

data flow analysis
Tracking of value constraints along nonexcluded paths through the code.

> NOTE
> Tracking can be performed intraprocedurally, with various assumptions made about what happens at function call boundaries, or interprocedurally, where values are tracked flowing into function calls (directly or indirectly) as arguments and flowing back out either as return values or indirectly through arguments.
> Data flow analysis may or may not track values flowing into or out of the heap or take into account global variables. When this specification refers to values flowing, the key point is contrast with variables or expressions because a given variable or expression may hold different values along different paths and a given value may be held by multiple variables or expressions along a path.

data race [ISO/IEC 9899:2011]
The execution of a program contains a data race if it contains two conflicting actions in different threads, at least one of which is not atomic, and neither happens before the other. Any such data race results in undefined behavior.

denial-of-service attack
Also *DoS attack*. An attempt to make a computer resource unavailable to its intended users.

diagnostic message [ISO/IEC 9899:2011]
A message belonging to an implementation-defined subset of the implementation's message output. A diagnostic message may indicate a constraint violation or a valid but questionable language construct. Messages typically include the file name and line number pointing to the offending code construct. In addition, implementations often indicate the severity of the problem. Although the C Standard does not specify any such requirement, the most severe problems often cause implementations to fail to fully translate a translation unit. Diagnostics output in such cases are termed errors. Other problems may cause implementations simply to issue a warning message and continue translating the rest of the program. See *error message* and *warning message*.

double-free vulnerability
An exploitable error resulting from the same allocated object being freed more than once.

error message
A diagnostic message generated when source code is encountered that prevents an implementation from translating a translation unit. See *diagnostic message* and *warning message*.

error tolerance [IEEE Std 610.12 1990]
The ability of a system or component to continue normal operation despite the presence of erroneous inputs.

exploit [ISO/IEC TS 17961:2013]
Technique that takes advantage of a security vulnerability to violate an explicit or implicit security policy.

fail safe [IEEE Std 610.12 1990]
Pertaining to a system or component that automatically places itself in a safe operating mode in the event of a failure—for example, a traffic light that reverts to blinking red in all directions when normal operation fails.

fail soft [IEEE Std 610.12 1990]
Pertaining to a system or component that continues to provide partial operational capability in the event of certain failures—for example, a traffic light that continues to alternate between red and green if the yellow light fails.

fatal diagnostic
A diagnostic message that causes an implementation not to perform the translation.

fault tolerance [IEEE Std 610.12 1990]
The ability of a system or component to continue normal operation despite the presence of hardware or software faults.

freestanding environment [ISO/IEC 9899:2011]
An environment in which C program execution may take place without any benefit of an operating system. Program startup might occur at some function other than `main()`, complex types might not be implemented, and only certain minimal library facilities are guaranteed to be available.

function-like macro [ISO/IEC 9899:2011]
A `#define` preprocessing directive that defines an identifier immediately followed by zero or more parameters, the ellipsis (`...`), or a combination of the two, enclosed in parentheses, similar syntactically to a function call. Subsequent instances of the macro name followed by a parenthesized list of arguments in a translation unit are replaced by the replacement list of preprocessing tokens that constitute the remainder of the directive. See *object-like macro* and *unsafe function-like macro*.

hosted environment [ISO/IEC 9899:2011]
An environment that is not freestanding. Program startup occurs at `main()`, complex types are implemented, and all C standard library facilities are available.

implementation [ISO/IEC 9899:2011]
Particular set of software, running in a particular translation environment under particular control options, that performs translation of programs for, and supports execution of functions in, a particular execution environment.

implementation-defined behavior [ISO/IEC 9899:2011]
Unspecified behavior whereby each implementation documents how the choice is made.

in-band error indicator [ISO/IEC 9899:2011]
A library function return value on error that can never be returned by a successful call to that library function.

incomplete type [ISO/IEC 9899:2011]
A type that describes an identifier but lacks information needed to determine the size of the identifier.

indeterminate value [ISO/IEC 9899:2011]
Either an unspecified value or a trap representation.

invalid pointer
A pointer that is not a valid pointer.

liveness
Every operation or method invocation executes to completion without interruptions, even if it goes against safety.

locale-specific behavior [ISO/IEC 9899:2011]
Behavior that depends on local conventions of nationality, culture, and language that each implementation documents.

lvalue [ISO/IEC 9899:2011]
An expression with an object type or an incomplete type other than `void`. The name *lvalue* comes originally from the assignment expression `E1 = E2`, in which the left operand `E1` is required to be a (modifiable) lvalue. It is perhaps better considered as representing an object "locator value."

mitigation [Seacord 2005a]
Methods, techniques, processes, tools, or runtime libraries that can prevent or limit exploits against vulnerabilities.

nonpersistent signal handler
Signal handler running on an implementation that requires the program to again register the signal handler after occurrences of the signal to catch subsequent occurrences of that signal.

normal program termination [IEEE Std 1003.1-2013]
Normal termination occurs by a return from `main()`, when requested with the `exit()`, `_exit()`, or `_Exit()` functions, or when the last thread in the process terminates by returning from its start function by calling the `pthread_exit()` function, or through cancellation. See *abnormal termination*.

object-like macro [ISO/IEC 9899:2011]
A `#define` preprocessing directive that defines an identifier with no parentheses. Subsequent instances of the macro name in a translation unit are replaced by the replacement list of preprocessing tokens that constitute the remainder of the directive. See *function-like macro*.

out-of-band error indicator [ISO/IEC TS 17961:2013]
A library function return value used to indicate nothing but the error status.

out-of-domain value [ISO/IEC TS 17961:2013]
One of a set of values that is not in the domain of a particular operator or function.

reentrant [ISO/IEC/IEEE 24765:2010]
Pertaining to a software module that can be entered as part of one process while also in execution as part of another process and still achieve the desired results.

reliability [IEEE Std 610.12 1990]
The ability of a system or component to perform its required functions under stated conditions for a specified period of time.

restricted sink [ISO/IEC 9899:2011]
Operands and arguments whose domain is a subset of the domain described by their types.

robustness [IEEE Std 610.12 1990]
The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

rvalue [ISO/IEC 9899:2011]
Value of an expression.

sanitize [ISO/IEC TS 17961:2013]
Assure by testing or replacement that a tainted or other value conforms to the constraints imposed by one or more restricted sinks into which it may flow.

> NOTE
> If the value does not conform, either the path is diverted to avoid using the value or a different, known-conforming value is substituted—for example, adding a null character to the end of a buffer before passing it as an argument to the `strlen` function.

security flaw [ISO/IEC TS 17961:2013]
Defect that poses a potential security risk.

security policy [Internet Society 2000]
Set of rules and practices that specify or regulate how a system or organization provides security services to protect sensitive and critical system resources.

sequence point [ISO/IEC 9899:2011]
Evaluation of an expression may produce side effects. At specific points in the execution sequence called *sequence points*, all side effects of previous evaluations have completed, and no side effects of subsequent evaluations have yet taken place.

side effect [ISO/IEC 9899:2011]
Changes in the state of the execution environment achieved by accessing a volatile object, modifying an object, modifying a file, or calling a function that does any of those operations.

> NOTE
> The IEC 60559 standard for binary floating-point arithmetic requires certain user-accessible status flags and control modes. Floating-point operations implicitly set the status flags; modes affect result values of floating-point operations. Implementations that support such a floating-point state are required to regard changes to it as side effects. These are detailed in Annex F of the C Standard.

static analysis [ISO/IEC TS 17961:2013]
Any process for assessing code without executing it.

strictly conforming [ISO/IEC 9899:2011]
A strictly conforming program is one that uses only those features of the language and library specified in the international standard. Strictly conforming programs are intended to be maximally portable among conforming implementations and cannot, for example, depend on implementation-defined behavior.

string [ISO/IEC 9899:2011]
A contiguous sequence of characters terminated by and including the first null character.

tainted source [ISO/IEC TS 17961:2013]
External source of untrusted data.

> NOTE
> Tainted sources include
>
> - parameters to the `main()` function
> - the returned values from `localeconv()`, `fgetc()`, `getc`, `getchar()`, `fgetwc()`, `getwc()`, and `getwchar()`
> - the strings produced by `getenv()`, `fscanf()`, `vfscanf()`, `vscanf()`, `fgets()`, `fread()`, `fwscanf()`, `vfwscanf()`, `vwscanf()`, `wscanf()`, and `fgetws()`

tainted value [ISO/IEC TS 17961:2013]
Value derived from a tainted source that has not been sanitized.

target implementation [ISO/IEC TS 17961:2013]
Implementation of the C programming language whose environmental limits and implementation-defined behavior are assumed by the analyzer during the analysis of a program.

TOCTOU, TOCTTOU
Time-of-check, time-of-use (TOCTOU), also referred to as time-of-check-to-time-of-use (TOCTTOU), represents a vulnerability in which access control checks are nonatomic with the operations they protect, allowing an attacker to violate access control rules.

trap representation [ISO/IEC 9899:2011]
Object representation that does not represent a value of the object type. Attempting to read the value of an object that has a trap representation other than by an expression that has a character type is undefined. Producing such a representation by a side effect that modifies all or any part of the object other than by an expression that has a character type is undefined.

undefined behavior (UB) [ISO/IEC 9899:2011]
Behavior, upon use of a nonportable or erroneous program construct or of erroneous data, for which the C Standard imposes no requirements. An example of undefined behavior is the behavior on integer overflow.

unexpected behavior
Well-defined behavior that may be unexpected or unanticipated by the programmer; incorrect programming assumptions.

unsafe function-like macro
A function-like macro whose expansion causes one or more of its arguments not to be evaluated exactly once.

unsigned integer wrapping
Computation involving unsigned operands whose result is reduced modulo the number that is one greater than the largest value that can be represented by the resulting type.

unspecified behavior [ISO/IEC 9899:2011]
Behavior for which the standard provides two or more possibilities and imposes no further requirements on which is chosen in any instance.

unspecified value [ISO/IEC 9899:2011]
A valid value of the relevant type where the C Standard imposes no requirements on which value is chosen in any instance. An unspecified value cannot be a trap representation.

untrusted data [ISO/IEC 11889-1:2009]
Data originating from outside of a trust boundary.

valid pointer [ISO/IEC TS 17961:2013]
Pointer that refers to an element within an array or one past the last element of an array. See *invalid pointer*.

> NOTE
> For the purposes of this definition, a pointer to an object that is not an element of an array behaves the same as a pointer to the first element of an array of length one with the type of the object as its element type. (See C Standard, 6.5.8, paragraph 4.)
> For the purposes of this definition, an object can be considered to be an array of a certain number of bytes; that number is the size of the object as produced by the `sizeof` operator. (See C Standard, 6.3.2.3, paragraph 7.)

validation [IEC 61508-4]
Confirmation by examination and provision of objective evidence that the particular requirements for a specific intended use are fulfilled.

verification [IEC 61508-4]
Confirmation by examination and provision of objective evidence that the requirements have been fulfilled.

vulnerability [ISO/IEC TS 17961:2013]
Set of conditions that allows an attacker to violate an explicit or implicit security policy.

warning message
A diagnostic message generated when source code is encountered that does not prevent an implementation from translating a translation unit. See *diagnostic message* and *error message*.