Amsterdam Compiler Kit-ANSI C compiler compliance statements

Hans van Eck

Dept. of Mathematics and Computer Science
Vrije Universiteit
Amsterdam, The Netherlands

This document specifies the implementation-defined behaviour of the ANSI-C front end of the Amsterdam Compiler Kit as required by ANS X3.159-1989. Since the implementation-defined behaviour sometimes depends on the machine compiling on or for, some items will be left unspecified in this document†. The compiler assumes that it runs on a UNIX system.

ANS A.6.3.1:

- Diagnostics are placed on the standard error output. They have the following specification:
  "<file>", line <nr>: [(<class>)] <diagnostic>
  There are three classes of diagnostics: "error", "strict" and "warning". When the class is "error", the <class> is absent.
  The class "strict" is used for violations of the standard which are not severe enough to stop compilation. An example is the the occurrence of non white-space after an '#else' or '#endif' pre-processing directive. The class "warning" is used for legal but dubious constructions. An example is overflow of constant expressions.

ANS A.6.3.2:

- The function 'main' can have two arguments. The first argument is an integer specifying the number of arguments on the command line. The second argument is a pointer to an array of pointers to the arguments (as strings).
- Interactive devices are terminals.

ANS A.6.3.3:

- The number of significant characters is an option. By default it is 64. There is a distinction between upper and lower case.

ANS A.6.3.4:

- The compiler assumes ASCII-characters in both the source and execution character set.
- There are no multi-byte characters.
- There 8 bits in a character.
- Character constants with values that can not be represented in 8 bits are truncated.
- Character constants that are more than 1 character wide will have the first character specified in the least significant byte.
- The only supported locale is "C".
- A plain 'char' has the same range of values as 'signed char'.

ANS A.6.3.5:

- The compiler assumes that it works on and compiles for a 2-complement binary-number system. Shorts will use 2 bytes and longs will use 4 bytes. The size of integers are machine dependent.

---

† when cross-compiling, run-time behaviour may be different from compile-time behaviour

- Converting an integer to a shorter signed integer is implemented by ignoring the high-order byte(s) of the former. Converting a unsigned integer to a signed integer of the same type is only done in administration. This means that the bit-pattern remains unchanged.
- The result of bitwise operations on signed integers are what can be expected on a 2-complement machine.
- If either operand is negative, whether the result of the / operator is the largest integer less than or equal to the algebraic quotient or the smallest integer greater than or equal to the algebraic quotient is machine dependent, as is the sign of the result of the % operator.
- The right-shift of a negative value is negative.

ANS A.6.3.6:

- The representation of floating-point values is machine-dependent. When native floating-point is not present an IEEE-emulation is used. The compiler uses high-precision floating-point for constant folding.
- Truncation is always to the nearest floating-point number that can be represented.

ANS A.6.3.7:

- The type returned by the sizeof-operator (also known as size_t) is 'unsigned int'. This is done for backward compatibility reasons.
- Casting an integer to a pointer or vice versa has no effect in bit-pattern when the sizes are equal. Otherwise the value will be truncated or zero-extended (depending on the direction of the conversion and the relative sizes).
- When a pointer is as large as an integer, the type of a 'ptrdiff_t' will be 'int'. Otherwise the type will be 'long'.

ANS A.6.3.8:

- Since the front end has only limited control over the registers, it can only make it more likely that variables that are declared as registers also end up in registers. The only things that can possibly be put into registers are : 'int', 'long', 'float', 'double', 'long double' and pointers.

ANS A.6.3.9:

- When a member of a union object is accessed using a member of a different type, the resulting value will usually be garbage. The compiler makes no effort to catch these errors.
- The alignment of types is a compile-time option. The alignment of a structure-member is the alignment of its type. Usually, the alignment is passed on to the compiler by the 'ack' program. When a user wants to do this manually, he/she should be prepared for trouble.
- A "plain" 'int' bit-field is taken as a 'signed int'. This means that a field with a size of 1 bit can only store the values 0 and -1.
- The order of allocation of bit-fields is a compile-time option. By default, high-order bits are allocated first.
- An enum has the same size as a "plain" 'int'.

ANS A.6.3.10:

- An access to a volatile declared variable is done by just mentioning the variable. E.g. the statement "x;" where x is declared volatile, constitutes an access.
- There is no fixed limit on the number of declarators that may modify an arithmetic, structure or union type, although specifying too many may cause the compiler to run out of memory.

ANS A.6.3.12:

- The maximum number of cases in a switch-statement is in the order of 1e9, although the compiler may run out of memory somewhat earlier.

ANS A.6.3.13:

- Since both the pre-processor and the compiler assume ASCII-characters, a single character constant in a conditional-inclusion directive matches the same value in the execution character set.
- The pre-processor recognizes -I... command-line options. The directories thus specified are searched first. After that, depending on the command that the preprocessor is called with, machine/system-dependant directories are searched. After that, ~em/include/_tail_ac and /usr/include are visited.
- Quoted names are first looked for in the directory in which the file which does the include resides.
- The characters in a h- or q- char-sequence are taken to be UNIX paths.
- Neither the compiler nor the preprocessor know any pragmas.
- Since the compiler runs on UNIX, __DATE__ and __TIME__ will always be defined.

ANS A.6.3.14:

- NULL is defined as ((void *)0). This in order to flag dubious constructions like "int x = NULL;".
- The diagnostic printed by 'assert' is as follows:
  "Assertion "<expr>" failed, file "<file>", line <line>",
  where <expr> is the argument to the assert macro, printed as string. (the <file> and <line> should be clear)
- The sets for character test macros.

name:	set:
isalnum()	0-9A-Za-z
isalpha()	A-Za-z
iscntrl()	\000-\037\177
islower()	a-z
isupper()	A-Z
isprint()	<space>-~ (== \040-\176)

  As an addition, there is an isascii() macro, which tests whether a character is an ascii character. Characters in the range from \000 to \177 are ascii characters.

- The behaviour of mathematic functions on domain error:

name:	returns:
asin()	0.0
acos()	0.0
atan2()	0.0
fmod()	0.0
log()	-HUGE_VAL
log10()	-HUGE_VAL
pow()	0.0
sqrt()	0.0

- Underflow range errors do not cause errno to be set.
- The function fmod() returns 0.0 and sets errno to EDOM when the second argument is 0.0.
- The set of signals for the signal() function depends on the UNIX-system which the compiler is compiling for. The default handling, semantics and behaviour of these signals are those specified by the operating system vendor. The default handling is not reset when SIGILL is received.
- A text-stream need not end in a new-line character.
- White space characters before a new-line appear when read in.

- There may be any number of null characters appended to a binary stream.
- The file position indicator of an append mode stream is initially positioned at the beginning of the file.
- A write on a text stream does not cause the associated file to be truncated beyond that point.
- The buffering intended by the standard is fully supported.
- A zero-length file actually exists.
- A file name can consist of any character, except for the '\0' and the '/'.
- A file can be open multiple times.
- When a remove() is done on an open file, reading and writing behave just as can be expected from a non-removed file. When the associated stream is closed, all written data will be lost.
- When a file exists prior to a call to rename(), the behaviour is that of the underlying UNIX system. Normally, the call would fail.
- The %p conversion in fprintf() has the same effect as %#x or %#lx, depending on the sizes of pointer and integer.
- The %p conversion in fscanf() has the same effect as %x or %lx, depending on the sizes of pointer and integer.
- A - character that is neither the first nor the last character in the scanlist for %[conversion is taken to be a range indicator. When the first character has a higher ASCII-value than the second, the - will just be put into the scanlist.
- The value of errno when fgetpos() or ftell() failed is that of lseek(). This means:

  EBADF – when the stream is not valid

  ESPIPE – when fildes is associated with a pipe (and on some systems: sockets)

  EINVAL – the resulting file pointer would be negative

- The messages generated by perror() depend on the value of errno. The mapping of errors to strings is done by strerror().
- When the requested size is zero, malloc(), calloc() and realloc() return a null-pointer.
- When abort() is called, output buffers will be flushed. Temporary files (made with the tmpfile() function) will have disappeared when SIGABRT is not caught or ignored.
- The exit() function returns the low-order eight bits of its argument to the environment.
- The predefined environment names are controlled by the user. Setting environment variables is done through the putenv() function. This function accepts a pointer to char as its argument. To set f.i. the environment variable TERM to a230 one writes

  ```
  putenv("TERM=a230");
  ```

  The argument to putenv() is stored in an internal table, so malloc'ed strings can not be freed until another call to putenv() (which sets the same environment variable) is made. The function returns 1 if it fails, 0 otherwise.

- The argument to system is passed as argument to /bin/sh -c.
- The strings returned by strerror() depend on errno in the following way:

errno	string
0	"Error 0",
EPERM	"Not owner",
ENOENT	"No such file or directory",
ESRCH	"No such process",
EINTR	"Interrupted system call",
EIO	"I/O error",
ENXIO	"No such device or address",

E2BIG	"Arg list too long",
ENOEXEC	"Exec format error",
EBADF	"Bad file number",
ECHILD	"No children",
EAGAIN	"No more processes",
ENOMEM	"Not enough core",
EACCES	"Permission denied",
EFAULT	"Bad address",
ENOTBLK	"Block device required",
EBUSY	"Mount device busy",
EEXIST	"File exists",
EXDEV	"Cross-device link",
ENODEV	"No such device",
ENOTDIR	"Not a directory",
EISDIR	"Is a directory",
EINVAL	"Invalid argument",
ENFILE	"File table overflow",
EMFILE	"Too many open files",
ENOTTY	"Not a typewriter",
ETXTBSY	"Text file busy",
EFBUG	"File too large",
ENOSPC	"No space left on device",
ESPIPE	"Illegal seek",
EROFS	"Read-only file system",
EMLINK	"Too many links",
EPIPE	"Broken pipe",
EDOM	"Math argument",
ERANGE	"Result too large"

everything else causes strerror() to return "unknown error"

- The local time zone is per default MET (GMT + 1:00:00). This can be changed through the TZ environment variable, or by some changes in the sources.
- The clock() function returns the number of ticks since process startup.

References

[1] ANS X3.159-1989 *American National Standard for Information Systems - Programming Language C*