NAME

c, **c78**, **c89**, **c90**, **c95**, **c99**, **c11**, **c17**, **c2x** - The C programming language

DESCRIPTION

C is a general purpose programming language, which has a strong connection with the UNIX operating system and its derivatives, since the vast majority of those systems were written in the C language. The C language contains some basic ideas from the BCPL language through the B language written by Ken Thompson in 1970 for the DEC PDP-7 machines. The development of the UNIX operating system was started on a PDP-7 machine in assembly language, but it made very difficult to port the existing code to other systems.

In 1972 Dennis M. Ritchie worked out the C programming language for further development of the UNIX operating system. The idea was to implement only the C compiler for different platforms, and implement most part of the operating system in the new programming language to simplify the portability between different architectures. It follows that C is very eligible for (but not limited to) writing operating systems and low-level applications.

The C language did not have a specification or standardized version for a long time. It went through a lot of changes and improvements for ages. In 1978, Brian W. Kernighan and Dennis M. Ritchie published the first book about C under the title "The C Programming Language". We can think of this book as the first specification of the language. This version is often referred as K&R C after the names of the authors. Sometimes it is referred as C78, as well, after the publishing year of the first edition of the book.

It is important to notice, that the instruction set of the language is limited to the most fundamental elements for simplicity. Handling of the standard I/O and such common functions are implemented in the libraries shipped with the compiler. As these functions are also widely used, it was demanded to include into the description what requisites the library should conform to, not just strictly the language itself. Accordingly, the aforementioned standards cover the library elements, as well. The elements of this standard library is still not enough for more complicated tasks. In this case the provided system calls of the given operating system can be used. To not lose the portability by using these system calls, the POSIX (Portable Operating System Interface) standard evolved. It describes what functions should be available to keep portability. Note, that POSIX is not a C standard, but an operating system standard and thus is beyond the scope of this manual. The standards discussed below are all C standards and only cover the C programming language and the accompanying library. All listed improvements for each standard edition are taken from the official standard drafts. For further details, check the publicly available drafts or purchase the published standards -- from either ISO or IEC resources.

After the publication of the book mentioned before, the American National Standards Institute (ANSI) started to work on standardizing the language, and they announced ANSI X3.159-1989 in 1989. It is

usually referred to as ANSI C or C89. The main difference in this standard were the function prototypes, which is a new way of declaring functions. With the old-style function declarations, the compiler was unable to check the sanity of the actual parameters at a function call. The old syntax was highly error-prone because incompatible parameters were hard to detect in the program code and the problem only showed up at run-time.

In 1990, the International Organization for Standardization (ISO) adopted the ANSI standard as ISO/IEC 9899:1990 in 1990. This is also referred to as ISO C or C90. It only contains negligible minor modifications against ANSI C, so the two standards often considered to be fully equivalent. This was a very important milestone in the history of the C language, but the development of the language did not stop.

The ISO C standard was later extended with an amendment as ISO/IEC 9899/AMD1 in 1995. This contained, for example, the wide-character support in <wchar.h> and <wctype.h>, and also restricted character set support via diagraphs and <iso646.h>. This amendment is usually referred to as C95. Two technical corrigenda were also published: Technical Corrigendum 1 as ISO/IEC 9899/COR1 in 1994 and Technical Corrigendum 2 as ISO/IEC 9899/COR2 in 1996. The continuous development and growth made it necessary to work out a new standard, which contains the new features and fixes the known defects and deficiencies of the language. As a result, ISO/IEC 9899:1999 was born in 1999 as the second edition of the standard. Similarly to the other standards, this is informally named after the publication year as C99. The improvements include (but are not limited to) the following:

- digraphs, trigraphs, and alternative spellings for the operators that use non-ISO646 characters in <iso646.h>
- extended multibyte and wide character library support in <wchar.h> and <wctype.h>
- variable length arrays
- flexible array members
- complex (and imaginary) number arithmetic support in <complex.h>
- type-generic math macros in <tgmath.h>
- the long long int type and library functions
- remove implicit int type
- universal character names (\u and \U)

- compound literals
- remove implicit function declaration
- BCPL style single-line comments
- allow mixed declarations and code
- the vscanf family of functions in <stdio.h> and <wchar.h>
- allow trailing comma in enum declaration
- inline functions
- the snprintf family of functions in <stdio.h>
- boolean type and macros in <stdbool.h>
- empty macro arguments
- _Pragma preprocessing operator
- __func__ predefined identifier
- va_copy macro in <stdarg.h>
- additional strftime conversion specifiers

Later in 2011, the third edition of the standard, ISO/IEC 1989:2011, commonly referred to as C11 (formerly C1x), came out and replaced the second edition by ISO/IEC 9899:1999/COR1:2001, ISO/IEC 9899:1999/COR2:2004, and ISO/IEC 9899:1999/COR3:2007. The improvements include (but are not limited to) the following:

- support for multiple threads of execution and atomic operations in <threads.h> and <stdatomic.h>
- additional floating-point characteristic macros in <float.h>
- querying and specifying alignment of objects in <stdalign.h> and <stdlib.h>

- Unicode character types and functions in <uchar.h>
- type-generic expressions
- static assertions in <assert.h>
- anonymous structures and unions
- remove the gets function from <stdio.h>
- add the aligned_alloc, at_quick_exit, and quick_exit functions in <stdlib.h>

C11 was later superseded by ISO/IEC 9899:2018, also known as C17 which was prepared in 2017 and published in June 2018 as the fourth edition. It incorporates the Technical Corrigendum 1 (ISO/IEC 9899:2011/COR1:2012) which was published in 2012. It addressed defects and deficiencies in C11 without introducing new features, only corrections and clarifications. Since there were no major changes in C17, the current standard for Programming Language C, is still considered C11 -- ISO/IEC 9899:2011, published 2011-12-08.

The next standard, the fifth, is currently referred to as C2x and is scheduled to be adopted by the end of 2021, with a publication date of 2022. When published, it will cancel and replace the fourth edition, ISO/IEC 9899:2018.

Some useful features have been provided as extensions by some compilers, but they cannot be considered as standard features.

ISO/IEC JTC1/SC22/WG14 committee is responsible for the ISO/IEC 9899, C Standard.

SEE ALSO

c89(1), c99(1), cc(1)

STANDARDS

ANSI, *X3.159-1989 (aka C89 or ANSI C)*.

ISO/IEC, *9899:1990 (aka C90)*.

ISO/IEC, *9899:1990/AMD 1:1995, Amendment 1: C Integrity (aka C95)*.

ISO/IEC, *9899:1990/COR 1:1994, Technical Corrigendum 1*.

ISO/IEC, *9899:1990/COR 2:1996, Technical Corrigendum 2*.

ISO/IEC, *9899:1999 (aka C99)*.

ISO/IEC, *9899:1999/COR 1:2001, Technical Corrigendum 1*.

ISO/IEC, *9899:1999/COR 2:2004, Technical Corrigendum 2*.

ISO/IEC, *9899:1999/COR 3:2007, Technical Corrigendum 3*.

ISO/IEC, *TR 24731-1:2007 (aka bounds-checking interfaces)*.

ISO/IEC, *TS 18037:2008 (aka, embedded C)*.

ISO/IEC, *TR 24747:2009 (aka mathematical special functions)*.

ISO/IEC, *TR 24732:2009 (aka decimal floating-point)*.

ISO/IEC, *TR 24731-2:2010 (aka dynamic allocation functions)*.

ISO/IEC, *9899:2011 (aka C11)*.

ISO/IEC, *9899:2011/COR 1:2012, Technical Corrigendum 1*.

ISO/IEC, *TS 17961:2013 (aka C secure coding rules)*.

ISO/IEC, *TS 18861-1:2014 (aka binary floating-point)*.

ISO/IEC, *TS 18861-2:2015 (aka decimal floating-point)*.

ISO/IEC, *TS 18861-3:2015 (aka interchange and extended types)*.

ISO/IEC, *TS 18861-4:2015 (aka supplementary functions)*.

ISO/IEC, *TS 17961:2013/COR 1:2016 (aka C secure coding rules TC1)*.

ISO/IEC, *TS 18861-5:2016 (aka supplementary attributes)*.

ISO/IEC, *9899:2018 (aka C17)*.

HISTORY

This manual page first appeared in FreeBSD 9.0.

AUTHORS

This manual page was originally written by Gabor Kovesdan <*gabor@FreeBSD.org*>. It was updated for FreeBSD 14.0 by Faraz Vahedi <*kfv@kfv.io*> with information about more recent C standards.