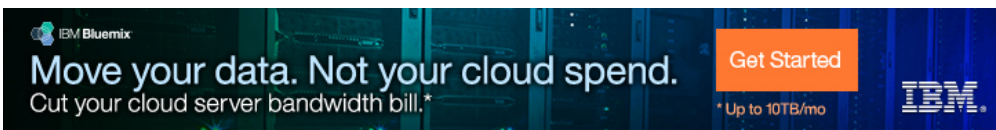


Join the Stack Overflow Community

Stack Overflow is a community of 7.0 million programmers, just like you, helping each other.
Join them; it only takes a minute:

[Sign up](#)

What are the most common naming conventions in C?



What are the naming conventions commonly use in C? I know there are at least two:

1. GNU / linux / K&R with lower_case_functions
2. ? name ? with UpperCaseFoo functions

I am talking about C only here. Most of our projects are small embedded systems in which we use C.

Here is the one I am planning on using for my next project:

C Naming Convention

```
Struct      TitleCase
Struct Members lower_case or lowerCase

Enum        ETitleCase
Enum Members ALL_CAPS or lowerCase

Public functions pfx_TitleCase (pfx = two or three letter module prefix)
Private functions TitleCase
Trivial variables i,x,n,f etc...
Local variables lower_case or lowerCase
Global variables g_lowerCase or g_lower_case (searchable by g_ prefix)
```

[c naming-conventions](#)

edited Nov 12 '09 at 13:41

asked Nov 12 '09 at 13:16



JeffV

15.3k 21 82 107

- 6 I wouldn't force a 'g_' prefix on global variables; I would enforce meaningful names (so client_locale and not cl_lc as a global variable name). Classic C doesn't use camel-case; I've written code in camel-case in C, and it looks weird (so I don't do it like that any more). That said, it isn't wrong - and consistency is more important than which convention is used. Avoid typedefs that encapsulate structure pointers; consider the C standard - 'FILE *' is spelled thus, not FILE_PTR. – [Jonathan Leffler](#) Nov 12 '09 at 13:34
- 1 @Jonathan Leffler, whats wrong with g_ to signify globals? In embedded systems I have had trouble before in which it was hard to track down inter-module dependencies through global vars and extern g_something. I personally think it is generally a bad idea, but this sort of thing usually gets done for performance reasons. For instance, a global flag that is set by an interrupt indicating that the data is ready. – [JeffV](#) Nov 12 '09 at 13:47
- 1 For what it's worth, this naming convention was mostly ripped from PalmOS API conventions. Also, it is similar to the convention used in O'Reilly's book: "Programming Embedded Systems with C and GNU Development Tools". Personally, I like the TitleCase in function names. I was thinking of going with lowerCamelCase in internal linkage functions (which I called private in my question). – [JeffV](#) Nov 12 '09 at 14:57
- 2 @Chris Lutz, I agree, whole heartedly. Wherever possible vars are to be kept at the narrowest scope. Note that there are actually three scopes we are discussing: local to a function, local to a module (no externs linkage to the variable) and the globals with external linkage. It is common to have "global to a module"

variables in embedded systems. Therefore, care must be taken to identify the globals with external linkage so they can be kept to a minimum and the module interactions understood. This is where the "g_" prefix is helpful. – [JeffV](#) Nov 13 '09 at 0:23

5 I like to prefix global variables with `//`. – [plafer](#) Nov 24 '15 at 5:26

7 Answers

The most important thing here is consistency. That said, I follow the GTK+ coding convention, which can be summarized as follows:

1. All macros and constants in caps: `MAX_BUFFER_SIZE` , `TRACKING_ID_PREFIX` .
2. Struct names and typedefs in camelcase: `GtkWidget` , `TrackingOrder` .
3. Functions that operate on structs: classic C style: `gtk_widget_show()` , `tracking_order_process()` .
4. Pointers: nothing fancy here: `GtkWidget *foo` , `TrackingOrder *bar` .
5. Global variables: just don't use global variables. They are evil.
6. Functions that are there, but shouldn't be called directly, or have obscure uses, or whatever: one or more underscores at the beginning: `_refrobnicate_data_tables()` , `_destroy_cache()` .

answered Nov 12 '09 at 14:22



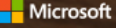

[axel_c](#)

4,890 19 39

7 In point six I prefer to use `static` and skip the module prefix, so if `gtk_widget_show()` was a function with file scope it would become simply `widget_show()` with static storage class added. – [August Karlstrom](#) Aug 15 '13 at 14:28

12 Additional note about point 6: the C standard has some rules about reserving names that begin with `_` for implementation and future use. There are a few exceptions to names beginning with `_` but in my opinion it is not worth the trouble to memorize. A safe rule to go by is to never use names beginning with `_` in your code. Relevant C FAQ entry: c-faq.com/~scs/cgi-bin/faqcat.cgi?sec=decl#namespace – [jw013](#) Mar 21 '14 at 21:08

1 for point six, what about using trailing underscore? or a prefix such as "internal_"? – [dragonxwang](#) May 4 '16 at 5:28

Got 5 minutes?
Then you have time to deploy
your first cloud solution.

Try Azure — free

"Struct pointers" aren't entities that need a naming convention clause to cover them. They're just `struct Whatever *`. DON'T hide the fact that there is a pointer involved with a clever and "obvious" typedef. It serves no purpose, is longer to type, and destroys the balance between declaration and access.

answered Nov 12 '09 at 13:18



[unwind](#)

273k 45 358 488

18 +1 for the "don't hide pointers" stuff - even though this answer doesn't address much of the rest of the question (yet). – [Jonathan Leffler](#) Nov 12 '09 at 13:35

@unwind, I tend to agree. However, sometimes a pointer is not intended to be external dereferenced and it is more of a handle to the consumer than an actual pointer to a struct they will be using. That's what I left the `TitleCasePtr` for. typedef struct { fields } MyStruct, *MyStructPtr; – [JeffV](#) Nov 12 '09 at 13:35

I am removing the `TitleCasePtr`, it is distracting from the actual question. – [JeffV](#) Nov 12 '09 at 13:40

-1 from me since a pointer type declaration reduces clutter, especially in function signatures and the "imbalance" between declaration and access only shows up in the implementation file - the client don't (shouldn't) access the field members directly. – [August Karlstrom](#) Aug 15 '13 at 14:47

@AugustKarlstrom Fine. I don't understand what's so "only" about the implementation file though, isn't that code, too? I didn't interpret the question as only being about "external" names. All code is "implementation" at some level. – [unwind](#) Aug 15 '13 at 14:59

Well firstly C doesn't have public/private/virtual functions. That's C++ and it has different conventions. In C typically you have:

- Constants in ALL_CAPS
- Underscores to delimit words in structs or function names, hardly ever do you see camel case in C;
- structs, typedefs, unions, members (of unions and structs) and enum values typically are in lower case (in my experience) rather than the C++/Java/C#/etc convention of making the first letter a capital but I guess it's possible in C too.

C++ is more complex. I've seen a real mix here. Camel case for class names or lowercase+underscores (camel case is more common in my experience). Structs are used rarely (and typically because a library requires them, otherwise you'd use classes).

answered Nov 12 '09 at 13:21



[cletus](#)

431k 123 790 884

@cletus, I realize that. By private I mean functions that are not exposed externally in the module header and are not intended to be used by code external to the module. Public would be module API functions intended to be used externally. – [JeffV](#) Nov 12 '09 at 13:28

3 You could regard static functions as private; the question doesn't mention virtual. But +1 for 'seldom see camel-case in C'. – [Jonathan Leffler](#) Nov 12 '09 at 13:30

2 I think Jeff meant `external linkage` for "public functions" and `internal linkage` for "private functions". – [pmg](#) Nov 12 '09 at 13:30

1 I've seen constants starting with a k as well as in: `kBufferSize`. Not sure where that comes from. – [JeffV](#) Nov 12 '09 at 14:14

1 ALL_CAPS is often used for enum values, too. – [caf](#) Nov 13 '09 at 1:15

I would recommend against mixing camel case and underscore separation (like you proposed for struct members). This is confusing. You'd think, hey I have `get_length` so I should probably have `make_subset` and then you find out it's actually `makeSubset`. Use the principle of least astonishment, and be consistent.

I do find CamelCase useful to type names, like structs, typedefs and enums. That's about all, though. For all the rest (function names, struct member names, etc.) I use underscore_separation.

answered Nov 12 '09 at 14:01



[Eli Bendersky](#)

131k 52 262 338

1 Yes, the main thing about any naming convention is predictability and consistency. Also, because the C library itself using all lowercase with `_` for spacing, I would recommend using that just so you don't have to deal with 2 different naming conventions in a project (assuming you don't write a wrapper around libc to make it conform to your naming.. but thats gross) – [Earlz](#) Nov 12 '09 at 14:14

It also uses typedefs with an "t" on the end, but I don't see anyone recommending that. In fact, the standard library is even inconsistent: `div_t (stdlib.h)` is a struct and so is `tm (time.h)`. Also, take a look at the `tm` struct members, they all are prefixed with `tm` which seems pointless and ugly (IMO). – [JeffV](#) Nov 12 '09 at 14:27

"I do find CamelCase useful to type names..." If you start it off capitalised, it's actually PascalCase. – [Tagc](#) Dec 15 '16 at 13:37

Coding in **C#, java, C, C++ and objective C** at the same time, I've adopted a very simple and clear naming convention to simplify my life.

First of all, it relies on the power of modern IDEs (such as eclipse, Xcode...), with the possibility to get fast information by hovering or ctrl click... Accepting that, I suppressed the use of any prefix, suffix and other markers that are simply given by the IDE.

Then, the convention:

- Any names MUST be a readable sentence explaining what you have. Like "this is my convention".
- Then, 4 methods to get a convention out of a sentence:
 1. **THIS_IS_MY_CONVENTION** for macros, enum members
 2. **ThisIsMyConvention** for file name, object name (class, struct, enum, union...), function name, method name, typedef
 3. **this_is_my_convention** global and local variables, parameters, struct and union elements

4. **thisismyconvention** [optional] very local and temporary variables (such like a for() loop index)

And that's it.

It gives

```
class MyClass {
    enum TheEnumeration {
        FIRST_ELEMENT,
        SECOND_ELEMENT,
    }

    int class_variable;

    int MyMethod(int first_param, int second_parameter) {
        int local_variable;
        TheEnumeration local_enum;
        for(int myindex=0, myindex<class_variable, myindex++) {
            localEnum = FIRST_ELEMENT;
        }
    }
}
```

edited Jan 8 '15 at 13:54

answered Jan 8 '15 at 13:38



Luc Fourestier

71 1 2

Here's an (apparently) uncommon one, which I've found useful: module name in CamelCase, then an underscore, then function or file-scope name in CamelCase. So for example:

```
Bluetooth_Init()
CommsHub_Update()
Serial_TxBuffer[]
```

answered Nov 12 '09 at 23:01



Steve Melnikoff

2,120 1 14 23

I'm confused by one thing: You're planning to create a new naming convention for a new project. Generally you should have a naming convention that is company- or team-wide. If you already have projects that have any form of naming convention, you should not change the convention for a new project. If the convention above is just codification of your existing practices, then you are golden. The more it differs from existing *de facto* standards the harder it will be to gain mindshare in the new standard.

About the only suggestion I would add is I've taken a liking to `_t` at the end of types in the style of `uint32_t` and `size_t`. It's very C-ish to me although some might complain it's just "reverse" Hungarian.

answered Nov 12 '09 at 14:27



jmucchiello

13.2k 5 31 55

Well, the conventions here are all over the place and inconsistent which is why I am setting out to document one. Also, it is why I am asking. To see what the community consensus is. – JeffV Nov 12 '09 at 14:31

I understand that pain. But there must be some subset of your existing conventions that is most popular. You should start there and not on a random Internet web page. Also you should ask your other devs what they would consider good. – jmucchiello Nov 12 '09 at 16:46

- 5 I believe type names ending in `_t` are reserved by the POSIX standard. – caf Nov 13 '09 at 1:18

`uint32_t` is a POSIX type? – jmucchiello Nov 13 '09 at 3:20

- 1 Name finishing with `_t` are reserved. See gnu.org/software/libc/manual/html_node/Reserved-Names.html, "Names that end with `'_t'` are reserved for additional type names." – Étienne Aug 13 '13 at 19:53