sign up log in tour help stack overflow careers

Stack Overflow is a question and answer site for professional and enthusiast programmers. It's 100% free, no registration required. Take the 2-minute tour ×

what is the difference between re-entrant function and recursive function in C?

In C I know about the recursive function but I heard about the re-entrant function.

What is that? And whats the difference between them?

language-agnostic reentrancy reentrant

edited Mar 25 '13 at 12:37
ulidtko
4,170 2 19 51

asked Nov 4 '08 at 9:17
Manoj Doubts
2,289 10 28 40

5 isn't this question language agnostic? – Rob Stevenson-Leggett Nov 4 '08 at 10:42

1 The wikipedia page is fairly useful. – brice Jun 11 '11 at 9:34

you can refer this also geeksforgeeks.org/archives/9096 – Student Arya Jun 24 '12 at 6:51

8 Answers

A function is re-entrant if it supports having multiple threads of execution "going through" it at the same time. This might be due to actual multi-threading, and I use this case below, or due to other things as pointed out by other posters. Multi-threading was the first that came to mind, and is perhaps also the easiest to understand, so I focused on that case.

This means that the function cannot use static "global" data, since that data would then be accessed by two (or more) threads in parallel, often breaking horribly. A re-entrant function often has an explicit argument to hold any call-specific state, rather than storing it statically.

`strtok()` is a classic case of a function in the C standard library that is well-known *not* to be re-entrant.

[Edit]: There are a bunch of insights, clarifications and corrections in the comments, so please read those as well! Thanks for the help, folks.

edited Nov 4 '08 at 10:15

answered Nov 4 '08 at 9:20
unwind
200k 30 270 387

yeap thank u very much – Manoj Doubts Nov 4 '08 at 9:23

1 reentrancy is not limited to concurrent systems (-1) – xtofl Nov 4 '08 at 9:42

3 Thread local storage only fixes re-entrance due to multithreading, not re-entrance due to recursion. – tolomea Nov 4 '08 at 9:58

3 yeah, -1 . reentrancy doesn't mean concurrency-safe. – Johannes Schaub - litb Nov 24 '08 at 16:51

2 -1. No concurrency required. Consider a function which modifies a shared buffer (runs in the same thread, but somehow calls itself again, directly or indirectly). – user166390 Dec 3 '10 at 4:48

It's easier to remember when you understand what the term means.

The term "re-entrant" means that it is safe to "**re-enter**" the function while it is already executed, typically in a concurrent environment.

In other words, when two tasks can execute the function at the same time without interfering with each other, then the function is re-entrant. A function is not re-entrant when the execution by one task has an impact on the influence of another task. This typically is the case when a global state or data is used. A function that uses only local variables and arguments is typically re-entrant.

answered Nov 4 '08 at 9:30

user34005
241 1 4

+1 for great points in a very simple and easy to understand words especially for beginners. – Jay D Jun 15 '12 at 23:19

"...typically in a concurrent environment." This is misleading. Reentrant and thread-safe are not the same thing: stackoverflow.com/questions/856823/threadsafe-vs-re-entrant – Hawkeye Parker Aug 6 '14 at 22:30

What unwind *originally* said is mostly correct - except that it is not limited to multi-threading (also, protecting global data with locks makes it thread safe - but *not* necessarily re-entrant). [Edit] He's fixed his post to account for this now :-)

A function may also be re-entered on the same thread as a result of recursion - either directly or indirectly (ie, function a calls function b which calls function c which calls function a).

Of course if you have protected against re-entrancy on the basis that multiple threads may call it then you are covered for the recursive cases too. That's not true the other way around, however.

edited Nov 4 '08 at 9:38

answered Nov 4 '08 at 9:29

philsquared
14k 6 48 80

2 If you use a thread local storage approach to solving re-entrance due to multithreading then it won't fix the re-entrance due to recursion case. – tolomea Nov 4 '08 at 9:57

That's true. I hope that my intention was clear - but you raise a valid caveat. – philsquared Nov 4 '08 at 10:38

"Re-entrance" of a function occurs when it is called before a previous invocation has returned. There are three main reasons for that to occur: recursion (the function calls itself), multi-threading and interruption. Recursion is normally easier, since it is clear that the function will be re-entered. Multi-threading and interruption are more tricky, as the re-entrance will be asynchronous. As stated in other answers, in most cases the function should not modify global data (reading global data is ok, some kings of writing is ok if protected as critical sections).

answered Nov 4 '08 at 10:33

Daniel Quadros

2 @Daniel Quadros: I wish I could vote this up more. A function which uses static data but backs it up to a pseudo-stack when calling any function which might call it (and then restores it after) is recursion safe but not interruption-safe. A function which uses static data but copies it to a pseudo-stack on entry and restores it on exit will be interruption-safe but not re-entrant. Re-entrant functions simply can't pretend static data is local to them. – supercat Jul 27 '10 at 23:23

Warning, Protecting a critical section with a lock will only make your function not re-entrant. If you were interrupted while holding the lock and the same function called, you'd get a deadlock. Only atomic, uninterruptible writes can be called if you want your function to be re-entrant, and even then, care should be taken with the logic (Ie, you're not implicitly keeping state in the ordering of reads and writes). – brice Jun 11 '11 at 9:26

Here is it:

- A reentrant function can be called simultaneously by multiple threads provided that each

invocation of the function references unique data.

- A thread-safe function can be called simultaneously by multiple threads when each invocation references shared data. All access to the shared data is serialized.

Shamelessly stolen from the Qt manual. But it's a short and concise definition. Basicially, a non-reentrant function is also not `recursion-safe`.

Now, what is a `recursive` function? It's a kind of definition of a function. Recursive function are defined in terms of themself. They reduce input, call theirself, until a basic case can be figured out without the need of calling theirself again.

So we have two things.

- recursive functions are a kind of definition.
- reentrant functions are functions that guarantee multiple threads can call them, provided each time unique data is accessed.

Now, the multiple-threads vehicle above serves only the purpose of having multiple activations of the function at the same time. But if you have a recursive function, you *also* have multiple activations of that functions at the same time. Most recursive functions therefor must be re-entrant too.

edited Nov 24 '08 at 17:12

answered Nov 24 '08 at 16:57


Johannes Schaub - litb
275k 56 561 939

A Re entrant function is a function which guaranteed that which can be work well under multi threaded environment. mean while function is access by one thread, another thread can call it... mean there is separate execution stack and handling for each... So function should not contain any static or shared variable which can harm or disturb the execution..

Mean function which can be called by thread, while running from another thread safely..... and properly.... hope I have answered the correct thing....

And of course rem that re-entrant function is not same as recursive function.... totally different concept.... A Re entrant function is a function which guaranteed that which can be work well under multi threaded environment. mean while function is access by one thread, another thread can call it... mean there is separate execution stack and handling for each... So function should not contain any static or shared variable which can harm or disturb the execution..

mean it should not contain any static or shared variable....

Mean function which can be called by thread, while running from another thread safely..... and properly.... hope I have answered the correct thing....

And of course rem that re-entrant function is not same as recursive function.... totally different concept....

Read more: http://wiki.answers.com/Q/What_is_a_reentrant_function#ixzz1wut38jLF Wiki : http://en.wikipedia.org/wiki/Reentrancy_%28computing%29

edited Jun 5 '12 at 14:33

answered Jun 5 '12 at 11:16


kapilddit
439 4 19

All re-entrant code is a recursion but not all recursion is a re-entrant. Example for recursion is, any function, which calls itself directly or indirectly. Example for re-entant is, interrupt handler routines.

answered Dec 3 '10 at 4:36

Virupakshi C M
11

All recursive code are re-entrant...but not all re-entrant code recursive.

answered Jan 27 '12 at 19:12

Yashoda
7 1

4 Not necessarily so. If a recursive function modifies global state, it may not be re-entrant from other threads. – Alexey Frunze Jan 27 '12 at 19:51

protected by Bo Persson Jan 27 '12 at 19:54

Thank you for your interest in this question. Because it has attracted low-quality answers, posting an answer now requires 10 reputation on this site.

Would you like to answer one of these unanswered questions instead?