Reentrant Function

Difficulty Level : Easy ● Last Updated : 26 Apr, 2018

A function is said to be reentrant if there is a provision to interrupt the function in the course of execution, service the interrupt service routine and then resume the earlier going on function, without hampering its earlier course of action. Reentrant functions are used in applications like hardware interrupt handling, recursion, etc.

The function has to satisfy certain conditions to be called as reentrant:
1. It may not use global and static data. Though there are no restrictions, but it is generally not advised. because the interrupt may change certain global values and resuming the course of action of the reentrant function with the new data may give undesired results.

2. It should not modify it's own code. This is important because the course of action of the function should remain the same throughout the code. But, this may be allowed in case the interrupt routine uses a local copy of the reentrant function every time it uses different values or before and after the interrupt.

Reentrancy is distinct from, but closely related to, thread-safety. A function can be thread-safe and still not reentrant. For example, a function could be wrapped all around with a mutex (which avoids problems in multithreading environments), but if that function is used in an interrupt service routine, it could starve waiting for the first execution to release the mutex. The key for avoiding confusion is that reentrant refers to only one thread executing. It is a concept from the time when no multitasking operating systems existed. (Source : https://en.wikipedia.org/wiki/Reentrancy_(computing))

Example of Non-Reentrant Functions:

```
// A non-reentrant example
// [The function depends on global variable i]

int i;

// Both fun1() and fun2() are not reentrant

// fun1() is NOT reentrant because it uses global variable i
int fun1()
{
    return i * 5;
}

// fun2() is NOT reentrant because it calls a non-reentrant
// function
int fun2()
{
   return fun1() * 5;
}
```

Example of Reentrant Functions:

In the below code, fun2 is a reentrant function. If an interrupt that pauses its execution and shifts the control to fun1. After fun1

```
// Both fun1() and fun2() are reentrant
int fun1(int i)
{
    return i * 5;
}

int fun2(int i)
{
   return fun1(i) * 5;
}
```

Article compiled by **Venki**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Previous

Critical Section in Synchronization

Next

Endian order and binary files

RECOMMENDED ARTICLES Page : 1 2

31, Jul 20

02 **PHP | gmp_setbit() Function**
11, Jun 18

03 **How to Call a C function in Python**
09, Nov 18

04 **getch() function in C with Examples**
04, Mar 20

06 **Difference between Preprocessor Directives and Function Templates in C++**
04, Jan 21

07 **Amazon EC2 - Creating an Elastic Cloud Compute Instance**
29, May 21

08 **Difference between Instance Variable and Local Variable**
26, May 21

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy | Normal | Medium | Hard | Expert

Improved By : VictorDev

Article Tags : Articles

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

GeeksforGeeks
5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org