Assignment

Day 6

Sunday, 10 September 2023

By: Ibrahim Tarek

SV G2 / Intake #3

Table of Contents

1. Executive Summary

This report discusses each of the following topics:

1- The difference between reentrant and non-reentrant functions.
2- The difference between synchronous and asynchronous functions.
3- What is call back function.

2. Reentrant vs. Non-Reentrant Function

2.1. Reentrant function

Function which if it is interrupted, then continues its execution after the ISR, it will behave right and normal.

2.2. Non-reentrant function

Function which if it is interrupted, then continues its execution after the ISR, it may not behave will.

2.3. Notes

2.3.1. Atomic access is the access which occurs in one clock cycle so no interrupt will affect it.

2.3.2. If the access is not atomic, problems may occur.

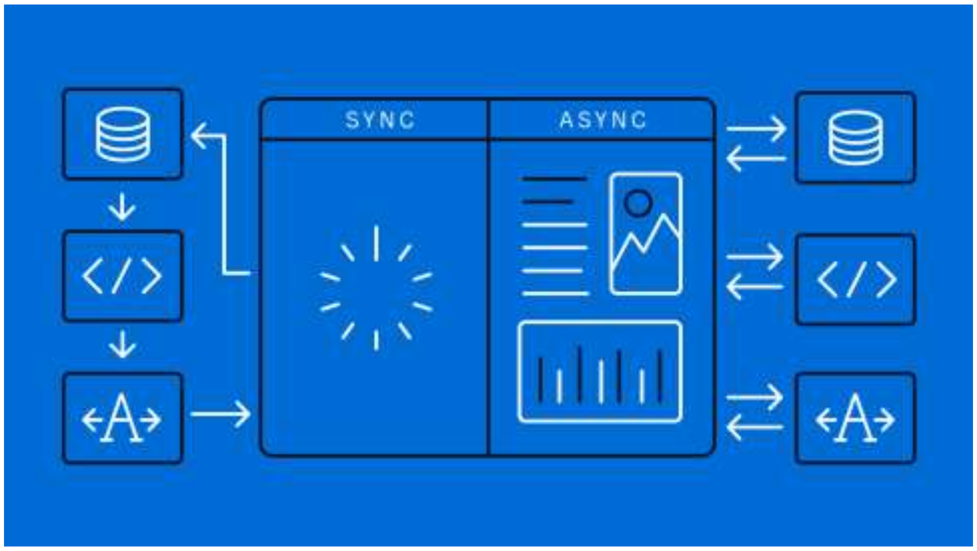
2.3.3. Any function access without protection statically allocated data is non-reentrant.

3. Sync. vs. Async. Function

Synchronous, sometimes referred to as “sync,” and asynchronous, also known as “async,” are two different types of programming models.

Understanding how these two models differ is critical in building application programming interfaces (APIs), creating event-based architectures, and deciding how to handle long-running tasks.

But before deciding which method to use and when, it’s important to know a few quick facts about synchronous programming and asynchronous programming.



3.1. Synchronous Function

A function in which instructions in synchronous code executes in a given sequence and each operation waits for the previous operation to complete its execution.

It starts the job and gets the result (maybe with waiting).

3.2. Asynchronous Function

A function in which instructions in asynchronous code can execute in parallel and next operation can occur while the previous operation is still getting processed.

It starts the job only and another function will complete the task (i.e. ISR).

4. Call Back Function

A callback function is any function that receives the reference of another function as the argument to call the function. We can call a callback function using the function pointers.

4.1. Example

```
#include<stdio.h>
void passFunction()
{
	printf("Function to be passed\n");
}

void callBackFunction(void (*ptr)())
{
	(*ptr) ();
}

int main()
{
	void (*ptr)() = &passFunction;
	callBackFunction(ptr);
	return 0;
}
```

4.2. Use case

In layered architecture projects, you need to call a function that is implemented in a high layer inside another function inside a low layer. In this case you may use three functions:

1- The first is the function which is needed to be called.
2- The second is the function which need to call the first function.
3- The third is a help function to copy the function pointer from the high level layer to the low level layer.

5. References

5.1. https://www.geeksforgeeks.org/reentrant-function/

5.2. https://www.mendix.com/blog/asynchronous-vs-synchronous-programming/

5.3. https://www.codingninjas.com/studio/library/callback-function-in-c