# Section 1 / `const`

In C++, a variable declared to be const cannot be altered. This concept only *partially* translates to assembly language in a manner which is enforced by the hardware. The remainder of the functionality of `const` is enforced at compile time and therefore, at the assembly language level, all bets are off.

## What Is Not Enforced

Given this short C++ snippet:

```
#include <stdio.h>                                                  // 1
                                                                    // 2
void Const1() {                                                     // 3
    const int foo = 9;                                              // 4
    printf("%d\n", foo);                                            // 5
}                                                                   // 6
```

it is the C++ compiler that enforces the immutable nature of `foo`. Looking at the assembly language that is produced from this C++ code, you will see that `foo` is implemented like any other variable. Once in assembly language you are behind the defenses of the compiler so to speak. You can modify a `const` local variable or parameter to your heart's content. Should you? That's another question. Will it cause harm? It depends.

## What IS Enforced

Let's examine this code:

```
        .global        main                                         // 1
        .align        2                                             // 2
                                                                    // 3
        .section    .rodata                                         // 4
rotypo: .asciz      "Rypo"          // Meant this to be "Typo"      // 5
fmt:    .asciz      "%s\n"          // Used to print the strings    // 6
                                                                    // 7
        .data                                                       // 8
rwtypo: .asciz      "Rypo"          // Meant this to be "Typo"      // 9
                                                                    // 10
        .text                                                       // 11
                                                                    // 12
main:   str         x30, [sp, -16]!                                 // 13
        // Try to fix rwtypo --- this will work                     // 14
        ldr         x1, =rwtypo                                     // 15
        mov         w2, 'T'                                         // 16
        strb        w2, [x1]                                        // 17
        ldr         x0, =fmt        // Notice I already loaded rwtypo   // 18
```

```
bl          printf                                                  // 19
                                                                    // 20
// Try to fix rotypo --- this will end in tears                     // 21
ldr         x1, =rotypo                                             // 22
mov         w2, 'T'                                                 // 23
strb        w2, [x1]                                                // 24
ldr         x0, =fmt        // Notice I already loaded rotypo       // 25
bl          printf                                                  // 26
                                                                    // 27
ldr         x30, [sp], 16                                           // 28
mov         w0, wzr                                                 // 29
ret                                                                 // 30
                                                                    // 31
.end                                                                // 32
                                                                    // 33
```

In this example, we are declaring a `const` *global variable*.

Notice `.section .rodata`. This assembler directive tells the linker to place what follows into memory that will be marked as read only. Any attempt to modify something marked as read only by the hardware will result in a crash.

`rwtypo` can be read and written as it is located after a `.data`. The string contains a typo - we meant to write "Typo" but we experienced a senior moment and what got entered was "Rypo". Could happen to anyone.

For some reason, we will fix "Rypo", turning it into "Typo" using code rather than simply editing the file.

The same is true for `rotypo`. Extra credit for being consistent.

However, `rotypo` is going to live in a region of memory marked as read only at the hardware level because it is defined after a `.section .rodata`.

Will we be able to correct it using code?

tl;dr Nope.

This program, when run, will produce the following output:

```
const > ./a.out
Typo
Segmentation fault (core dumped)
const >
```

This demonstrates that the first attempt at modifying `rwtypo` worked. And then, an attempt to modify `rotypo` causes a segmentation fault.

## Summary

The meaning and function of `const` only partially translates to assembly language. `const` local variables and `const` parameters are just like any other data to assembly language. The constant nature of `const` local variables and parameters is implemented solely in the compiler.

`const` globals are made constant by the hardware. Attempting to modify a variable protected in this manner will be like poking a dragon. Best not to poke dragons.