IDATT2503 - Exercise 5 - Edvard Berdal Eek

30.09.2025

Task 1

To follow the prerequisites described in the repo I created a simple Ubuntu docker container with the **assembly-example** repo mounted in which i completed the task.

Change the output to “Hello World from Trondheim!”:

I first tried to just edit the text in the data section of the assembly code.

```
27 section .data                ; This section is for declaring initialized data
26   msg: db "Hello World from Trondheim!", 10 ; EDIT: Add "from Trondheim" to data
```

But I quickly realized that I had to increase the listed length of the message from 13 to 28 as it now was 15 characters longer and the output was cropped without it increased.

```
3   mov rdx, 28                 ; The length of message - EDIT: Increase with 15
```

This message is to be written three times to standard error (instead of standard output):

To achieve this I had to set the counter to 3 instead of 10.

```
13 _start:                       ; Execution begins here
12   mov rcx, 3                 ; Set counter to 10 - EDIT: Set counter to 3
```

And to make it be written to standard error I changed the file descriptor from 1 to 2.

```
7   mov rdi, 2                  ; File descriptor 1 - standard output - EDIT: change to 2 (stderr)
```

Verify that the program outputs to standard error instead of standard output:

To verify that the output was written to standard error, not standard output, I redirected the standard output and error from the program to specific files so that it could be examined after execution.

```
./hello 1> stdout.txt 2> stderr.txt
```

So when i later inspected the **stderr.txt** file I could see that it was all written there and that the **stdout.txt** file was empty.

```
root@fc732369c225:/workspace# cat stderr.txt
Hello World from Trondheim!
Hello World from Trondheim!
Hello World from Trondheim!
```

The program is to return an error code:

To return an error code, I set the the return code to 1.

```
26    mov rdi, 1                ; Exit with return code of 0 (no error) - EDIT: Change to 1
```

To verify that the program returns an error code, I executed the program and printed the previous exit status:

```
root@fc732369c225:/workspace# ./hello
Hello World from Trondheim!
Hello World from Trondheim!
Hello World from Trondheim!
root@fc732369c225:/workspace# echo $?
1
```

Task 2

In C, C++ and Rust write a function that takes a string as input, and returns a new string equal to the input but where &, < and > is replaced respectively with &, < and >

See attached files (**main.rs**, **main.cpp** and **main.c**).

Write examples with outputs where you use this function in the main() functions in the various programming languages

The output is the same in all the various programming languages:

```
Ampersand:
Original: What the he&&y
Altered: What the he&amp;&amp;y

Less than:
Original: What the he<<y
Altered: What the he&lt;&lt;y

Greater than:
Original: What the he>>y
Altered: What the he&gt;&gt;y
```

Task 3

Follow the instructions here to make a dynamic library and to reference this library from the executable c_example

I created a similar Ubuntu container as in task one only this time with **gcc** aswell. Then i follows the instructions in the repo.

```
root@4c1af4746ba2:/workspace# gcc -c -fPIC a_function.c more_functions.c
files for the shared library
root@4c1af4746ba2:/workspace# gcc -shared a_function.o more_functions.o -o libfunctions.so
tions.so from the object files
root@4c1af4746ba2:/workspace# cp libfunctions.so /usr/lib
```

In the function another_function there is a typing error. Fix this, update the dynamic library containing this file, and run c_example again (without recreating c_example after you fixed another_function). What is the result?

The result is that without having to recompile **c_example** it outputs the updated version of the function in the dynamic library.

```
root@4c1af4746ba2:/workspace# ./c_example
message: Hello World
message from a_function: Hello World
FIXED: You have called another_function
```