

# Assignment 4

IDATT2503 - Security in programming and cryptography  
Fall 2023

Francesca Grimaldi

## 1 Task 1

### 1.1 Modifications

First of all, I changed the message from `Hello World!` to `Hello World from Trondheim!`. I updated the length of the message and set the counter to 3, instead of 10. In order to write the message to standard error, I modified the file descriptor from 1 (standard output) to 2 (standard error). I then modified the line of code before the `syscall` to return an error code (1).

### 1.2 Verification

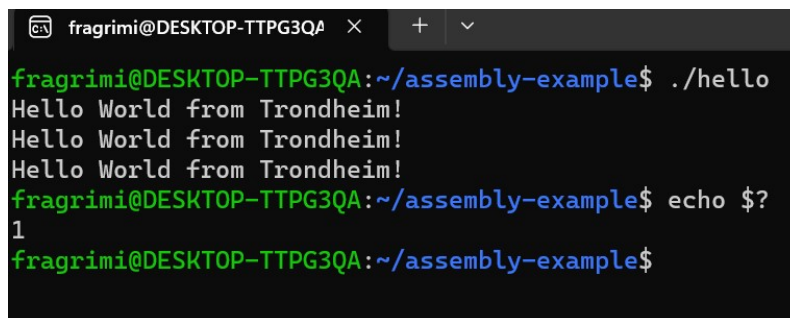
To test whether the program actually writes to standard error instead of standard output, I ran the following commands

```
./hello 1> hello_stdout.txt  
./hello 2> hello_stderr.txt
```

and checked the output to make sure that the `stdout` file was empty, whereas the `stderr` file contained the 3 sentences.

To verify the return of the error code, after the execution of the program I ran the command

```
echo $?
```

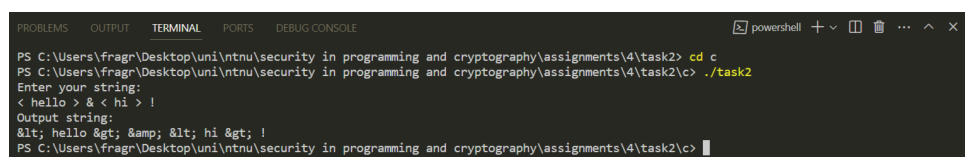


```
fragrimi@DESKTOP-TTPG3QA:~/assembly-example$ ./hello  
Hello World from Trondheim!  
Hello World from Trondheim!  
Hello World from Trondheim!  
fragrimi@DESKTOP-TTPG3QA:~/assembly-example$ echo $?  
1  
fragrimi@DESKTOP-TTPG3QA:~/assembly-example$
```

Figure 1: Error code in Task 1

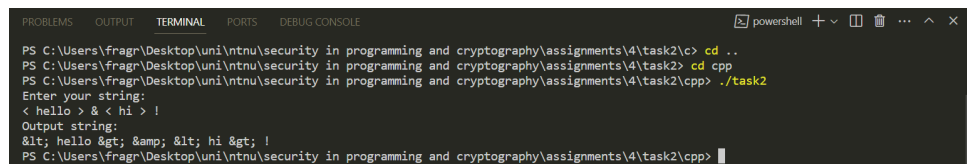
## 2 Task 2

The outputs of the three programs written respectively in C, C++ and Rust, are the following:



```
PS C:\Users\fragr\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2> cd c  
PS C:\Users\fragr\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2> ./task2  
Enter your string:  
< hello > & < hi > !  
Output string:  
&lt; hello &gt; &amp; &lt; hi &gt; !  
PS C:\Users\fragr\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2>
```

Figure 2: Output of C program

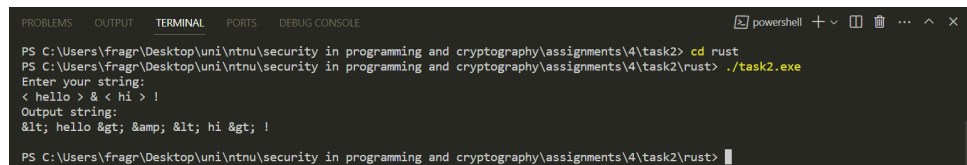


```

PS C:\Users\fragn\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2> cd ..
PS C:\Users\fragn\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2> cd cpp
PS C:\Users\fragn\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2\cpp> ./task2
Enter your string:
< hello > & < hi > !
Output string:
&lt; hello &gt; &amp; &lt; hi &gt; !
PS C:\Users\fragn\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2\cpp>

```

Figure 3: Output of C++ program



```

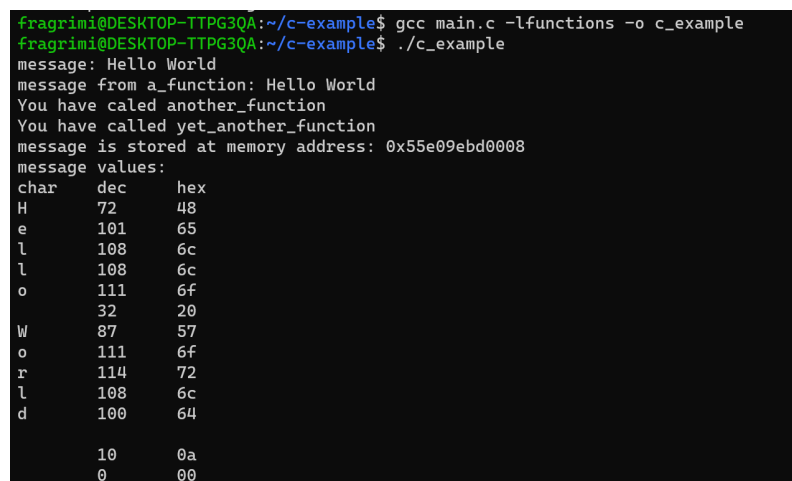
PS C:\Users\fragn\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2> cd rust
PS C:\Users\fragn\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2\rust> ./task2.exe
Enter your string:
< hello > & < hi > !
Output string:
&lt; hello &gt; &amp; &lt; hi &gt; !
PS C:\Users\fragn\Desktop\uni\ntnu\security in programming and cryptography\assignments\4\task2\rust>

```

Figure 4: Output of Rust program

### 3 Task 3

After creating the dynamic library and linking it to the executable file, this is the output:



```

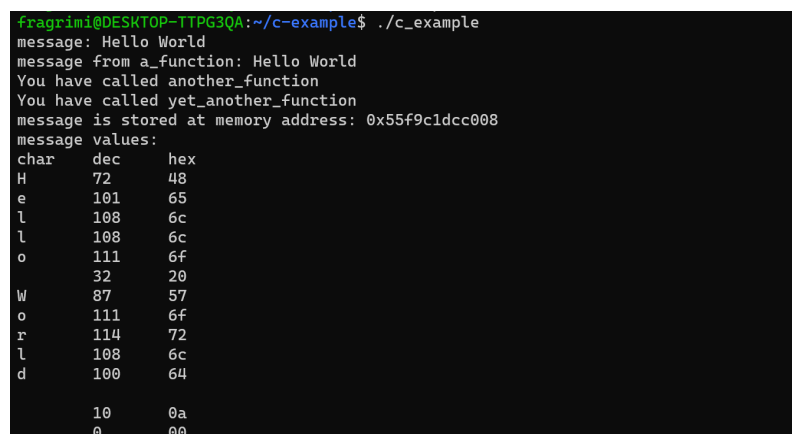
fragrimi@DESKTOP-TTPG3QA:~/c-example$ gcc main.c -lfunctions -o c_example
fragrimi@DESKTOP-TTPG3QA:~/c-example$ ./c_example
message: Hello World
message from a_function: Hello World
You have caled another_function
You have called yet_another_function
message is stored at memory address: 0x55e09ebd0008
message values:
char  dec  hex
H      72   48
e     101   65
l     108   6c
l     108   6c
o     111   6f
      32    20
W      87   57
o     111   6f
r     114   72
l     108   6c
d     100   64

      10    0a
      0     00

```

Figure 5: Output with spelling error

It is clear that there is a spelling error in the function named `another_function` ("caled"). Fixing it and updating the dynamic library produces the following result when running `c_example` again:



```

fragrimi@DESKTOP-TTPG3QA:~/c-example$ ./c_example
message: Hello World
message from a_function: Hello World
You have called another_function
You have called yet_another_function
message is stored at memory address: 0x55f9c1dcc008
message values:
char  dec  hex
H      72   48
e     101   65
l     108   6c
l     108   6c
o     111   6f
      32    20
W      87   57
o     111   6f
r     114   72
l     108   6c
d     100   64

      10    0a
      0     00

```

Figure 6: Output with fixed typo

To see the changes, there is no need to recompile `c_example`.