

Exercises static

static variables

1.

Compile and execute the file static-int-01.c. Try to understand the function `nr_of_calls` (it's not rocket science!!).

2.

Compile and execute the file static-int-02.c. Try to understand the function `nr_of_calls` (differs a bit from the previous one).

Why does the counter variable keep the value?

3.

Compile and execute the file static-int-03.c. Try to understand the function `nr_of_calls` (differs a bit from the previous one). None of the variables are static, still the variables (seem to) keep their value between calls. Why?

static functions

Compile the following C files into a program:

- static-funs-01.c
- motor.c

Execute the program to get a feeling of it.

4.

In the Simple Control Loop in the main function (`main.c`) you should add a `printf` statement like this:

```
fprintf(stdout, "Sneaky deaky: %d\n",
        existing_sneaky_little_function());
```

Compile again without using any flags/options such as `-Wall -Werror`. You can call a function not in the corresponding header file (`motor.h`). Why?

5.

Compile the files again, but now with the compilation flags/options `-Wall -Werror -pedantic`

It doesn't work. Why?

6.

Remove the `printf` statement. Instead you should (to the loop in `main.c`) add a call to the static function `doors_locked_intern` (located in the file `motor.c`).

It doesn't work. Why?

7.

You can, however, call the function `doors_locked` which in turn calls some static functions. It seems as if calling a static function directly doesn't work and an indirect call seems to work? Try out for yourself.

Read the following section for some more information about static functions:
https://en.wikibooks.org/wiki/C_Programming/Procedures_and_functions#Static_functions

Exercises const

1.

Compile the file `const-int-01.c` to make sure it works.

Change the value of the variable `i` on a line after the initialisation and recompile. Does it work? Why?

2.

Declare a pointer to `int` variable. Call the variable `harold` (after the famous author), Assign the variable `harold` the address of `i`.

Assign 129 to the memory address `harold` points to - you need to dereference `harold` to do this. Does the compile and produce a binary?

4.

We shall now change the assignment to `harold` a bit. Typecast the address of `i` like this:

```
(int*) (void*)&i;
```

Does it work? Why?