



# MY\_PRINTF

PRINTF COMMAND-LIKE



# MY\_PRINTF



**binary name:** libmy.a

**language:** C

**compilation:** via Makefile, including re, clean and fclean rules



- ✓ The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.
- ✓ All the bonus files (including a potential specific Makefile) should be in a directory named bonus.
- ✓ Error messages have to be written on the error output.

You must recode the **printf** function from the C library according to the C99 standard. Your function should be prototyped as the printf function.

You do not have to implement the C library printf buffer handling.



man 3 printf / man 3 stdarg

You must process all **printf** flags.



You must submit a Makefile that will create a library named my, as well as all source files. The libmy.a library must contain the my\_printf function, in addition to any other functions required to make it functional.



The whole libC is forbidden, except **va\_start**, **va\_end**, **va\_arg**, **malloc**, **free** and **write**.

## Makefile

In your `Makefile` you must list all of the `c` files separately (no `*.c`), compile them, and use the result to generate the `libmy.a` library.

Your `Makefile` must have the following rules:

- ✓ `libmy.a`
- ✓ `all` (which calls the `libmy.a` rule)
- ✓ `clean`
- ✓ `fclean` (which calls the `clean` rule)
- ✓ `re` (which calls the `fclean` and `libmy.a` rules)
- ✓ `unit_tests` (which calls the `fclean` and `libmy.a` rules, and then links the `lib` with the tests) - **Optional**
- ✓ `tests_run` (which calls the `unit_tests` rule and executes the `unit_tests` bin) - **Optional**

## Unit tests



Criterion includes mechanisms to test standard output and standard error, [you can learn more about it there...](#)

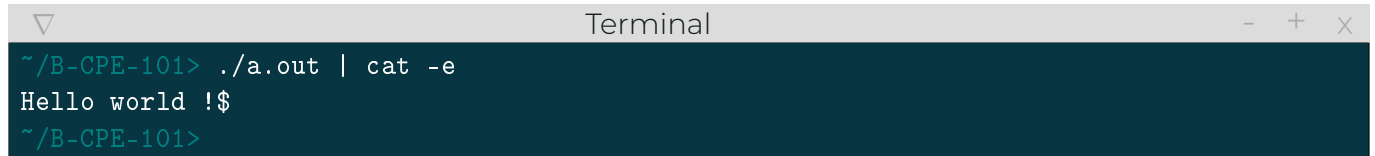
```
#include <riterion/criterion.h>
#include <riterion/redirect.h>
#include "my.h"

void redirect_all_std(void)
{
    cr_redirect_stdout();
    cr_redirect_stderr();
}

Test(my_printf, simple_string, .init = redirect_all_std)
{
    my_printf("hello world");
    cr_assert_stdout_eq_str("hello world");
}
```

## Examples

```
char str[8];  
  
my_strcpy(str, "world !");  
my_printf("Hello %s\n", str);
```

A terminal window titled "Terminal" with standard window controls (minimize, maximize, close). The prompt is ~/B-CPE-101>. The user enters ./a.out | cat -e, and the output is Hello world !\$. The prompt returns to ~/B-CPE-101>.

```
~/B-CPE-101> ./a.out | cat -e  
Hello world !$  
~/B-CPE-101>
```

## Bonuses

Be creative, you might also look for every versions of printf like sprintf, fprintf, dprintf, etc... %S and %b are considered as bonuses.



Every bonuses have to be in the bonus directory

{EPITECH}

