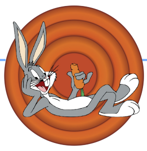

C - Pool - Tek1

Subject Day 07

C Pool Managers
looneytunes@epitech.eu

Contents

Instructions

- The subject may change until one hour before turn-in.
- Respect the norm takes time, but is good for you. This way your code will respect the norm since the first written line.
- Ask yourself if it's relevant to let a `main()` function in your turn-in knowing we will add our own.
- The robot will test your turn-in as follows:

```
$> cd ex_01
$> cc *.c -c
$> cc *.o ~moulinette/main_ex_01.o -L../lib/my/ -o ex01 -lmy
$> ./ex01
$> [...]
```

- This is a turn-in directory, of course you will only keep in it your final work revision. No temporary file should stand there!
  You shall leave in your directory no other files than those explicitly specified by the exercises.
  If one of your files prevents the compilation with `*.c`, the robot will not be able to do the correction and you will have a `0`. That is why it's in your interest to remove any file that doesn't work.

> For every exercise, except the one in which you build your library, your files shall not contain any of the function present in the library.

- Don't forget to discuss about it in the pool section of the forum !
- Turn-in directory:
  `Piscine_C_J07`

> *Hints* Remember it is always better to create your repository at the beginning of the day and to turn-in your work on a regular basis

> *Hints* On the instructions of each exercises, this directory is specified for every turn-in path

Unit Tests

- It is highly recommended to test your functions when you are developing them.
- Usually, it is common to create a function named "`main`" (and a dedicated file to host it) to check the functions separately.
- Create a directory named "`tests`".
- Create a function "`int main()`" in a file named "`tests-exercise_name.c`", stored inside the directory "`tests`" previously created.
- According to you, this function must contains all the necessary call to "`exercise_name`" to cover all possible cases (special or regular) of the function.

Indices

```
Here is a partial list of tests:
- Check the empty strings
- Check the pointer's values
```

Exercise 1 - libmy.a

- Build your own library in `Piscine_C_J07/lib/my/` and name it `libmy.a`.
- This library shall contain all the following functions:

```c
void my_putchar(char c);
int my_isneg(int nb);
int my_put_nbr(int nb);
int my_swap(int *a, int *b);
int my_putstr(char *str);
int my_strlen(char *str);
int my_getnbr(char *str);
void my_sort_int_tab(int *tab, int size);
int my_power_rec(int nb, int power);
int my_square_root(int nb);
int my_is_prime(int nombre);
int my_find_prime_sup(int nb);
char *my_strcpy(char *dest, char *src);
char *my_strncpy(char *dest, char *src, int nb);
char *my_revstr(char *str);
char *my_strstr(char *str, char *to_find);
int my_strcmp(char *s1, char *s2);
int my_strncmp(char *s1, char *s2, int nb);
char *my_strupcase(char *str);
char *my_strlowcase(char *str);
char *my_strcapitalize(char *str);
int my_str_isalpha(char *str);
int my_str_isnum(char *str);
int my_str_islower(char *str);
int my_str_isupper(char *str);
int my_str_isprintable(char *str);
int my_showstr(char *str);
int my_showmem(char *str, int size);
char *my_strcat(char *dest, char *src);
char *my_strncat(char *dest, char *src, int nb);
```

- Your `libmy.a` library shall imperatively be in the correct folder because it will be used to compile all your programs. From now on, you shall not have a single function present in your library in any of your sources.
- Turn-in:
  `Piscine_C_J07/lib/my/libmy.a`

Today you shall also code two functions, those from the exercises below, and include them in your library

Exercise 2 - my_strcat

- Write a function that copies a string after another (look at the man)
- The function shall be prototyped as follows:

```c
char *my_strcat(char *dest, char *src);
```

- Turn-in:
  `Piscine_C_J07/ex_02/my_strcat.c`

Hints `man strcat`

Exercise 3 - my_strncat

- Write a function that copies `n` characters from a string after another
- The function shall be prototyped as follows:

```c
char *my_strncat(char *dest, char *src, int nb);
```

- Turn-in:
  `Piscine_C_J07/ex_03/my_strncat.c`

Hints `man strncat`

Exercise 4 - my_aff_params

- Write a program that displays the received arguments (from the command line).
- We are talking about a program, that's why you shall have a `main` function in a `.c` of your turn-in repository.
- You shall display all the arguments, that includes argv[0].
- All the arguments shall be displayed on different lines.
- Turn-in directory:
  `Piscine_C_J07/ex_04/my_aff_params/`
- Example :

```
$>./a.out test "This is a test " retest | cat -e
./a.out$
test$
This is a test $
retest$
$>
```

Exercise 5 - my_rev_params

- Write a program that displays all the arguments received on the command line in the reverse order.
- You shall display all the arguments, that includes argv[0].
- All the arguments shall be displayed on different lines.
- Turn-in directory:
  `Piscine_C_J07/ex_05/my_rev_params/`
- Example :

```
$>./a.out test "This is a test " retest | cat -e
retest$
This is a test $
test$
./a.out$
$>
```

Exercise 6 - my_sort_params

- Write a program that displays all the arguments received on the command line ordered by ascii order.
- You shall display all the arguments, that includes argv[0].
- All the arguments shall be displayed on different lines.
- Turn-in directory:
  `Piscine_C_J07/ex_06/my_sort_params/`
- Example :

```
$>./a.out test "This is a test " retest | cat -e
./a.out$
This is a test $
retest$
test$
$>
```

That's all Folks!