C - Pool - Tek1

Subject Day 06

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.
- We will compile your files with the command `cc *.c`, adding our `main.c` and our `my_putchar.c` :

```
$> cd ex\_01
$> cc *.c ~moulinette/main_ex_01.c ~moulinette/my_putchar.c -o ex01
$> ./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.
- You are only allowed to use the `my_putchar` function to do the following exercises. This function will be provided, so:
  - You shall not have a `my_putchar.c` file in your turn-in directory.
  - The function `my_putchar` shall not be in any of your turned-in files.
- Don't forget to discuss about it in the pool section of the forum !
- Almost all of these functions exist in the "string" library. To get a full explanation on how a function works, you just need to use `man`.

Hints `For my_strcpy : man strcpy`

- Obviously, none of those functions shall contain a function from the "string" library
- Turn-in directory:
  `Piscine_C_J06`

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
- Pay attention to the possible values of an int
(0, min and max)
```

Exercise 1 - author

- The directory `Piscine_C_J06` shall contain a file named `auteur` with your login followed by a `'\n'`.
- Example :

```
foo_b@moulinette> cat -e auteur
foo_b$
foo_b@moulinette>
```

Exercise 2 - my_strcpy

- Write a function that copies a string into another. The destination string will already have enough memory to copy the source string.
- The function shall be prototyped as follows:

```
1   char *my_strcpy(char *dest, char *src);
```

- It shall return `dest`.
- Turn-in:
  `Piscine_C_J06/ex_02/my_strcpy.c`

Hints `Hint: man strcpy`

Exercise 3 - my_strncpy

- Write a function that copies `n` characters from a string into another. The destination string will already have sufficient memory to contain `n` characters.
  Add `'\0' if n > the length of the string`.
  Don't add `'\0' if n < the length of the string` (because `dest` is not supposed to contain more than `n` bytes).
- The function shall be prototyped as follows:

```
1   char *my_strncpy(char *dest, char *src, int n);
```

- It shall return `dest`.
- Turn-in:
  `Piscine_C_J06/ex_03/my_strncpy.c`

Hints `Hint: man strncpy`

Exercise 4 - my_revstr

- Write a function that reverse a string.
- The function shall be prototyped as follows:

```
1   char *my_revstr(char *str);
```

- It shall return `str`.
- Turn-in:
  `Piscine_C_J06/ex_04/my_revstr.c`

Exercise 5 - my_strstr

- Reproduce the behavior of the function `strstr`.
- The function shall be prototyped as follows:

```
1   char *my_strstr(char *str, char *to_find);
```

- Turn-in:
  `Piscine_C_J06/ex_05/my_strstr.c`

Hints `Take a look at the my_strcmp and my_strncmp functions`

Exercise 6 - my_strcmp

- Reproduce the behavior of the function `strcmp`.
- The function shall be prototyped as follows:

```
1   int my_strcmp(char *s1, char *s2);
```

- It shall return the same values as strcmp(3)
- Turn-in:
  `Piscine_C_J06/ex_06/my_strcmp.c`

Hints `man 3 strcmp`

Exercise 7 - my_strncmp

- Reproduce the behavior of the function `strncmp`.
- The function shall be prototyped as follows:

```
1   int my_strncmp(char *s1, char *s2, int n);
```

- It shall return the same values as strncmp(3)
- Turn-in:
  `Piscine_C_J06/ex_07/my_strncmp.c`

Exercise 8 - my_strupcase

- Write a function that sets in uppercase every letter of every word.
- The function shall be prototyped as follows:

```
1    char *my_strupcase(char *str);
```

- It shall return `str`.
- Turn-in:
`Piscine_C_J06/ex_08/my_strupcase.c`

Exercise 9 - my_strlowcase

- Write a function that sets in lowercase every letter of every word.
- The function shall be prototyped as follows:

```
1    char *my_strlowcase(char *str);
```

- It shall return `str`.
- Turn-in:
`Piscine_C_J06/ex_09/my_strlowcase.c`

Exercise 10 - my_strcapitalize

- Write a function that capitalizes each word.
- The function shall be prototyped as follows:

```
1   char *my_strcapitalize(char *str);
```

- It shall return `str`
- Turn-in:
  `Piscine_C_J06/ex_10/my_strcapitalize.c`

Hints `"hey, how are you ?  42words forty-two; fifty+one" gives "Hey, How Are You ?  42words Forty-Two; Fifty+One"`

Exercise 11 - my_str_isalpha

- Write a function that returns 1 if the string passed as parameter contains only alphabetical characters and returns 0 if the string contains another type of characters.
- The function shall be prototyped as follows:

```
1   int  my_str_isalpha(char *str);
```

- It shall return 1 if `str` is an empty string.
- Turn-in:
`Piscine_C_J06/ex_11/my_str_isalpha.c`

Exercise 12 - my_str_isnum

- Write a function that returns 1 if the string passed as parameter contains only digits and that returns 0 otherwise
- The function shall be prototyped as follows:

```
1   int my_str_isnum(char *str);
```

- It shall return 1 if `str` is an empty string.
- Turn-in:
`Piscine_C_J06/ex_12/my_str_isnum.c`

Exercise 13 - my_str_islower

- Write a function that returns 1 if the string passed as parameter only contains lowercase alphabetical characters and that returns 0 otherwise.
- The function shall be prototyped as follows:

```
1   int my_str_islower(char *str);
```

- It shall return 1 if `str` is an empty string
- Turn-in:
  `Piscine_C_J06/ex_13/my_str_islower.c`

Exercise 14 - my_str_isupper

- Write a function that returns 1 if the string passed as parameter only contains uppercase alphabetical characters and that returns 0 otherwise.
- The function shall be prototyped as follows:

```
1   int my_str_isupper(char *str);
```

- It should return 1 if `str` is an empty string.
- Turn-in:
`Piscine_C_J06/ex_14/my_str_isupper.c`

Exercise 15 - my_str_isprintable

- Write a function that returns 1 if the string passed as parameter only contains printable characters and that returns 0 otherwise.
- The function shall be prototyped as follows:

```
1   int my_str_isprintable(char *str);
```

- It should return 1 if `str` is an empty string.
- Turn-in:
`Piscine_C_J06/ex_15/my_str_isprintable.c`

Hints `man isprint`

Exercise 16 - my_putnbr_base

- Write a function that prints a number on the screen, in a given base.
- This number is supplied as an `int`, and the base is provided as a `string`
- The base contains all symbols that can be used to print the number:
  - `0123456789` is the decimal base commonly used to represent our numbers
  - `01` is a binary base
  - `0123456789ABCDEF` is an hexadecimal base
- The function shall return the number passed as parameter.
- The function shall handle negative numbers.
- The function shall be prototyped as follows:

```
1   int my_putnbr_base(int nbr, char *base);
```

- Turn-in:
  `Piscine_C_J06/ex_16/my_putnbr_base.c`

Exercise 17 - my_getnbr_base

- Write a function that returns a number.
- This number is provided as a `string`.
- The string expresses the number in a particular base, which is passed as second parameter.
- The function shall handle negative numbers.
- The function shall be able to manage several + or - following each others before the number.
- If a parameter contains an error, then the function returns 0.
  - str is an empty string
  - the base is empty
  - str contains characters which are not into the base and that are not + or -
  - the base contains a character more than one time
  - ...
- The function shall be prototyped as follows:

```
1   int my_getnbr_base(char *str, char *base);
```

- Turn-in:
  `Piscine_C_J06/ex_17/my_getnbr_base.c`

Exercise 18 - my_showstr

- Write a function that prints a string on screen. If this string contains non-printable characters, they shall be printed as hexadecimal (in lowercase) with a "backslash" before the value.
- The function shall be prototyped as follows:

```
1   int my_showstr(char *str);
```

- The function always returns 0.
- Turn-in:
  `Piscine_C_J06/ex_18/my_showstr.c`

Hints `my_showstr("I like\nponies !") prints "I like\0aponies !"`

Exercise 19 - my_showmem

- Write a function that prints a memory area on the screen.
- The output is divided into three columns:
  - The hexadecimal address of the first character of the line.
  - The content in hexadecimal.
  - The content in printable characters.
- If a character is not printable, it shall be replaced by a dot.
- Each line shall handle 16 characters.

```
?>./my_showmem
00000000: 6865 7920 6775 7973 2073 686f 7720 6d65  hey guys show me
00000010: 6d20 6973 2063 6f6f 6c20 796f 7520 6361  m is cool you ca
00000020: 6e20 646f 2073 6f6d 6520 7072 6574 7479  n do some pretty
00000030: 206e 6561 7420 7374 7566 6600 0f1b 7f05   neat stuff.....
00000040: 2e00 0102 0304 0506 0708 090e 0f1b 7f    ..............
?>./my_showmem | cat -te
00000000: 6865 7920 6775 7973 2073 686f 7720 6d65  hey guys show me$
00000010: 6d20 6973 2063 6f6f 6c20 796f 7520 6361  m is cool you ca$
00000020: 6e20 646f 2073 6f6d 6520 7072 6574 7479  n do some pretty$
00000030: 206e 6561 7420 7374 7566 6600 0f1b 7f05   neat stuff.....$
00000040: 2e00 0102 0304 0506 0708 090e 0f1b 7f    ..............$
?>
```

- This function always returns 0.
- The function shall be prototyped as follows:

```
1   int my_showmem(char *str, int size);
```

- Turn-in:
  `Piscine_C_J06/ex_19/my_showmem.c`

Hints Warning: Don't forget the padding if there are not enough characters to have a valid alignment