

C Piscine
Day 05

Staff 42 pedago@42.fr

Summary: This document is the subject for Day05 of the C Piscine @ 42.

Contents

1	Instructions		3
II	Foreword		5
III	Exercise 00 : ft_	_putstr	6
IV	Exercise 01 : ft_	_putnbr	7
\mathbf{V}	Exercise 02 : ft_	_atoi	8
VI	Exercise 03 : ft_	_strcpy	9
VII	Exercise 04: ft_	_strncpy	10
VIII	Exercise 05: ft_	_strstr	11
IX	Exercise 06: ft_	_strcmp	12
\mathbf{X}	Exercise 07: ft_	_strncmp	13
XI	Exercise 08: ft_	_strupcase	14
XII	Exercise 09 : ft_	_strlowcase	15
XIII	Exercise 10 : ft_	_strcapitalize	16
XIV	Exercise 11 : ft_	_strisalpha	17
XV	Exercise 12: ft_	_strisnumeric	18
XVI	Exercise 13: ft_	_strislowercase	19
XVII	Exercise 14: ft_	_strisuppercase	20
XVIII	Exercise 15: ft_	_strisprintable	21
XIX	Exercise 16: ft_	_strcat	22
XX	Exercise 17: ft_	_strncat	23
XXI	Exercise 18: ft_	_strlcat	24
XXII	Exercise 19: ft_	_strlcpy	25
XXIII	Exercise 20: ft	putnbr base	26

	C Piscine		Day 05
	XXIV Exercise 21 : ft_atoi_base	se	28
	XXV Exercise 22 : ft_putstr_:	non_printable	30
	XXVI Exercise 23 : ft_print_m	nemory	31
4			
4 /			
1			
		2	

Chapter I

Instructions

- Only this page will serve as reference: do not trust rumors.
- Watch out! This document could potentially change up to an hour before submission.
- Make sure you have the appropriate permissions on your files and directories.
- You have to follow the submission procedures for all your exercises.
- Your exercises will be checked and graded by your fellow classmates.
- On top of that, your exercises will be checked and graded by a program called Moulinette.
- Moulinette is very meticulous and strict in its evaluation of your work. It is entirely automated and there is no way to negotiate with it. So if you want to avoid bad surprises, be as thorough as possible.
- Moulinette is not very open-minded. It won't try and understand your code if it doesn't respect the Norm. Moulinette relies on a program called Norminator to check if your files respect the norm. TL;DR: it would be idiotic to submit a piece of work that doesn't pass Norminator's check.
- These exercises are carefully laid out by order of difficulty from easiest to hardest. We will not take into account a successfully completed harder exercise if an easier one is not perfectly functional.
- Using a forbidden function is considered cheating. Cheaters get -42, and this grade is non-negotiable.
- If ft_putchar() is an authorized function, we will compile your code with our ft_putchar.c.
- You'll only have to submit a main() function if we ask for a program.

- Moulinette compiles with these flags: -Wall -Wextra -Werror, and uses gcc.
- If your program doesn't compile, you'll get 0.
- ullet You <u>cannot</u> leave <u>any</u> additional file in your directory than those specified in the subject.
- Got a question? Ask your peer on your right. Otherwise, try your peer on your left.
- Your reference guide is called Google / man / the Internet /
- Check out the "C Piscine" part of the forum on the intranet.
- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...
- By Odin, by Thor! Use your brain!!!



Norminator must be launched with the $\mbox{-R CheckForbiddenSourceHeader}$ flag. Moulinette will use it too.

Chapter II

Foreword

Here is a discuss extract from the Silicon Valley serie:

- I mean, why not just use Vim over Emacs? (CHUCKLES)
- I do use Vim over Emac.
- Oh, God, help us! Okay, uh you know what? I just don't think this is going to work. Uh, I mean like, what, we're going to bring kids into this world with that over the That's not really fair to them, don't you think?
- Kids? We haven't even slept together.
- And guess what, it's never going to happen now, because there is no way I'm going t
- Richard! (PRESS SPACE BAR MANY TIMES)
- Wow. Okay. Goodbye.
- One tab saves you eight spaces! (DOOR SLAMS) (BANGING)

. . .

(RICHARD MOANS)

- Oh, my God! Richard, what happened?
- I just tried to go down the stairs eight steps at a time.
 I'm okay, though.
- See you around, Richard.
- Just making a point.

Hopefully, you are not forced to use emacs and your space bar to complete the following exercices.

Chapter III

Exercise 00: ft_putstr

	Exercise 00	
/	${ m ft_putstr}$	
Turn-in directory: ex	00/	
Files to turn in: ft_putstr.c		
Allowed functions: ft_putchar		

- 42 Classics: Theses exercises are key assignments that do not earn points, but are mandatory to validate in order to access to the real assignments of the day.
 - \bullet Create a function that displays a string of characters on the standard output.
 - Here's how it should be prototyped :

void ft_putstr(char *str);

Chapter IV

Exercise 01: ft_putnbr

	Exercise 01	
/	ft_putnbr	
Turn-in directory : ex01/		
Files to turn in : ft_putn	br.c	
Allowed functions: ft_pu	itchar	

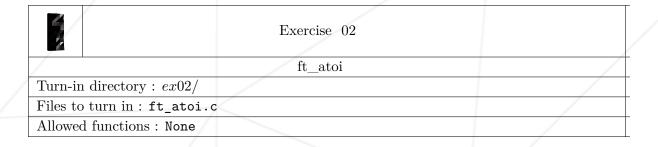
- 42 Classics: Theses exercises are key assignments that do not earn points, but are mandatory to validate in order to access to the real assignments of the day.
 - Create a function that displays the number entered as a parameter. The function has to be able to display all possible values within an int type variable.
 - Here's how it should be prototyped :

void ft_putnbr(int nb);

- For example:
 - o ft_putnbr(42) displays "42".

Chapter V

Exercise 02: ft_atoi



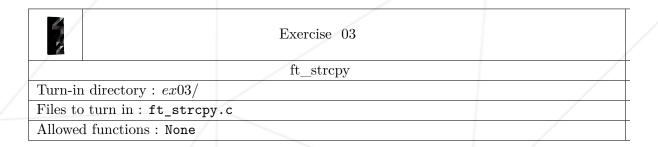
42 - Classics : Theses exercises are key assignments that do not earn points, but are mandatory to validate in order to access to the real assignments of the day.

- Reproduce the behavior of the function atoi (man atoi).
- Here's how it should be prototyped:

int ft_atoi(char *str);

Chapter VI

Exercise 03: ft_strcpy

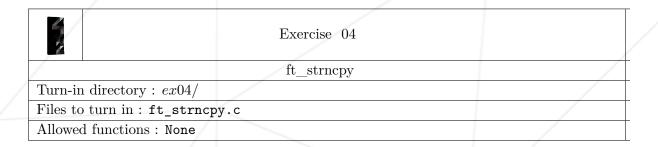


- Reproduce the behavior of the function strcpy (man strcpy).
- Here's how it should be prototyped :

char *ft_strcpy(char *dest, char *src);

Chapter VII

Exercise 04: ft_strncpy

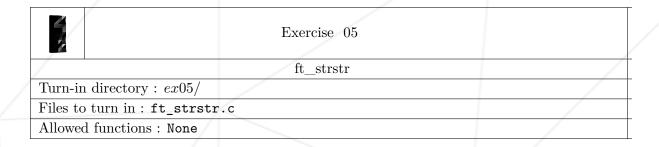


- Reproduce the behavior of the function strncpy (man strncpy).
- Here's how it should be prototyped :

char *ft_strncpy(char *dest, char *src, unsigned int n);

Chapter VIII

Exercise 05 : ft_strstr

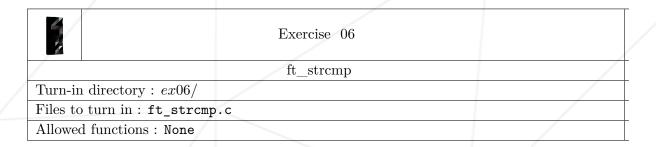


- Reproduce the behavior of the function strstr (man strstr).
- Here's how it should be prototyped :

char *ft_strstr(char *str, char *to_find);

Chapter IX

Exercise 06: ft_strcmp

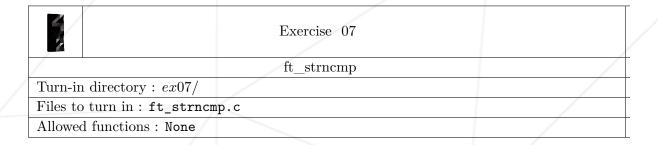


- Reproduce the behavior of the function strcmp (man strcmp).
- Here's how it should be prototyped :

int ft_strcmp(char *s1, char *s2);

Chapter X

Exercise 07: ft_strncmp

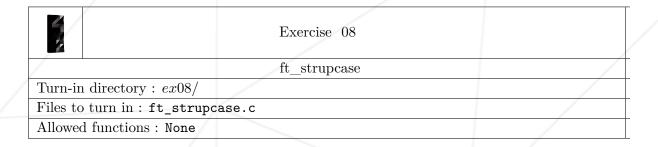


- Reproduce the behavior of the function strncmp (man strncmp).
- Here's how it should be prototyped :

int ft_strncmp(char *s1, char *s2, unsigned int n);

Chapter XI

Exercise 08: ft_strupcase



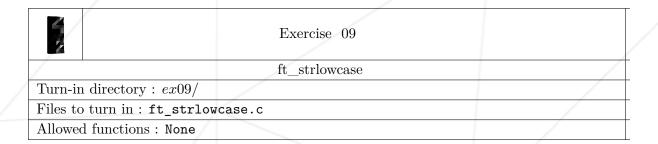
- Create a function that transforms every letter of every word to uppercase.
- Here's how it should be prototyped :

char *ft_strupcase(char *str);

• It should return str.

Chapter XII

Exercise 09: ft_strlowcase



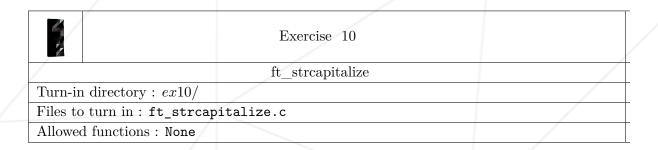
- Create a function that transforms every letter of every word to lowercase.
- Here's how it should be prototyped :

char *ft_strlowcase(char *str);

• It should return str.

Chapter XIII

Exercise 10: ft_strcapitalize



- Create a function that capitalizes the first letter of each word and transforms all other letters to lowercase.
- A word is a string of alphanumeric characters.
- Here's how it should be prototyped:

```
char *ft_strcapitalize(char *str);
```

- It should return str.
- For example:

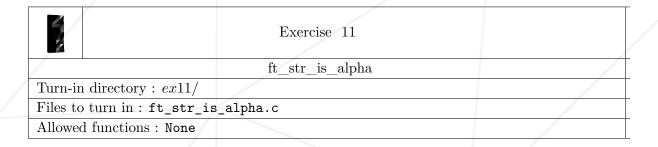
```
salut, comment tu vas ? 42mots quarante-deux; cinquante+et+un
```

• Becomes:

Salut, Comment Tu Vas ? 42mots Quarante-Deux; Cinquante+Et+Un

Chapter XIV

Exercise 11: ft_str_is_alpha

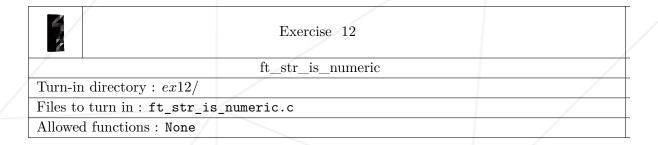


- Create a function that returns 1 if the string given as a parameter contains only alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

```
int ft_str_is_alpha(char *str);
```

Chapter XV

Exercise 12: ft_str_is_numeric

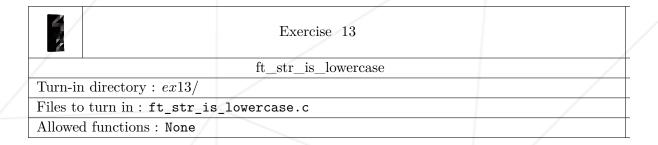


- Create a function that returns 1 if the string given as a parameter contains only digits, and 0 if it contains any other character.
- Here's how it should be prototyped:

int ft_str_is_numeric(char *str);

Chapter XVI

Exercise 13: ft_str_is_lowercase

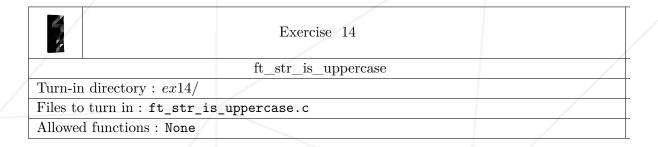


- Create a function that returns 1 if the string given as a parameter contains only lowercase alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped:

```
int ft_str_is_lowercase(char *str);
```

Chapter XVII

Exercise 14: ft_str_is_uppercase

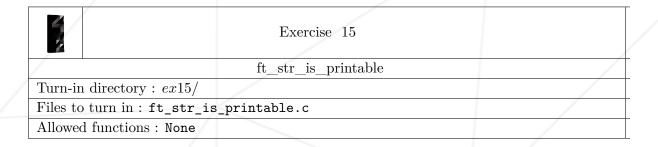


- Create a function that returns 1 if the string given as a parameter contains only uppercase alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

```
int ft_str_is_uppercase(char *str);
```

Chapter XVIII

Exercise 15: ft_str_is_printable

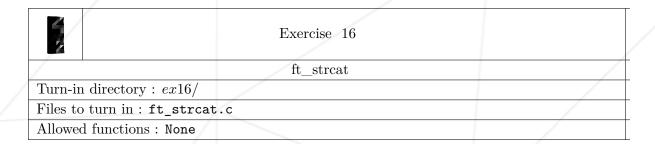


- Create a function that returns 1 if the string given as a parameter contains only printable characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

```
int ft_str_is_printable(char *str);
```

Chapter XIX

Exercise 16: ft_strcat

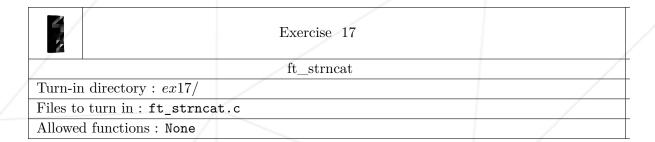


- Reproduce the behavior of the function strcat (man strcat).
- Here's how it should be prototyped :

char *ft_strcat(char *dest, char *src);

Chapter XX

Exercise 17: ft_strncat

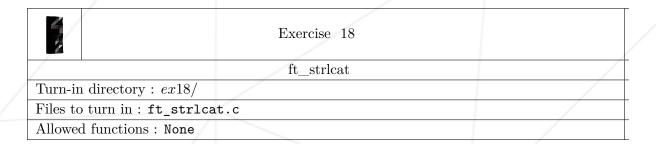


- Reproduce the behavior of the function strncat (man strncat).
- Here's how it should be prototyped :

char *ft_strncat(char *dest, char *src, int nb);

Chapter XXI

Exercise 18: ft_strlcat

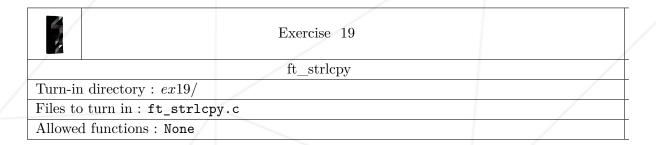


- Reproduce the behavior of the function strlcat (man strlcat).
- Here's how it should be prototyped :

unsigned int ft_strlcat(char *dest, char *src, unsigned int size);

Chapter XXII

Exercise 19: ft_strlcpy



- Reproduce the behavior of the function strlcpy (man strlcpy).
- Here's how it should be prototyped :

unsigned int ft_strlcpy(char *dest, char *src, unsigned int size);

Chapter XXIII

Exercise 20: ft_putnbr_base

	Exercise 20	
/	ft_putnbr_base	
Turn-in directory : $ex20/$		
Files to turn in : ft_putnbr_base.c		
Allowed functions: ft_put	char	

- Create a function that displays a number in a base system onscreen.
- This number is given in the shape of an int, and the radix in the shape of a string of characters.
- The base-system contains all useable symbols to display that number :
 - \circ 0123456789 is the commonly used base system to represent decimal numbers ;
 - 01 is a binary base system;
 - 0123456789ABCDEF an hexadecimal base system;
 - o poneyvif is an octal base system.
- The function must handle negative numbers.
- If there's an invalid argument, nothing should be displayed. Examples of invalid arguments:
 - base is empty or size of 1;
 - base contains the same character twice;
 - \circ base contains + or -;

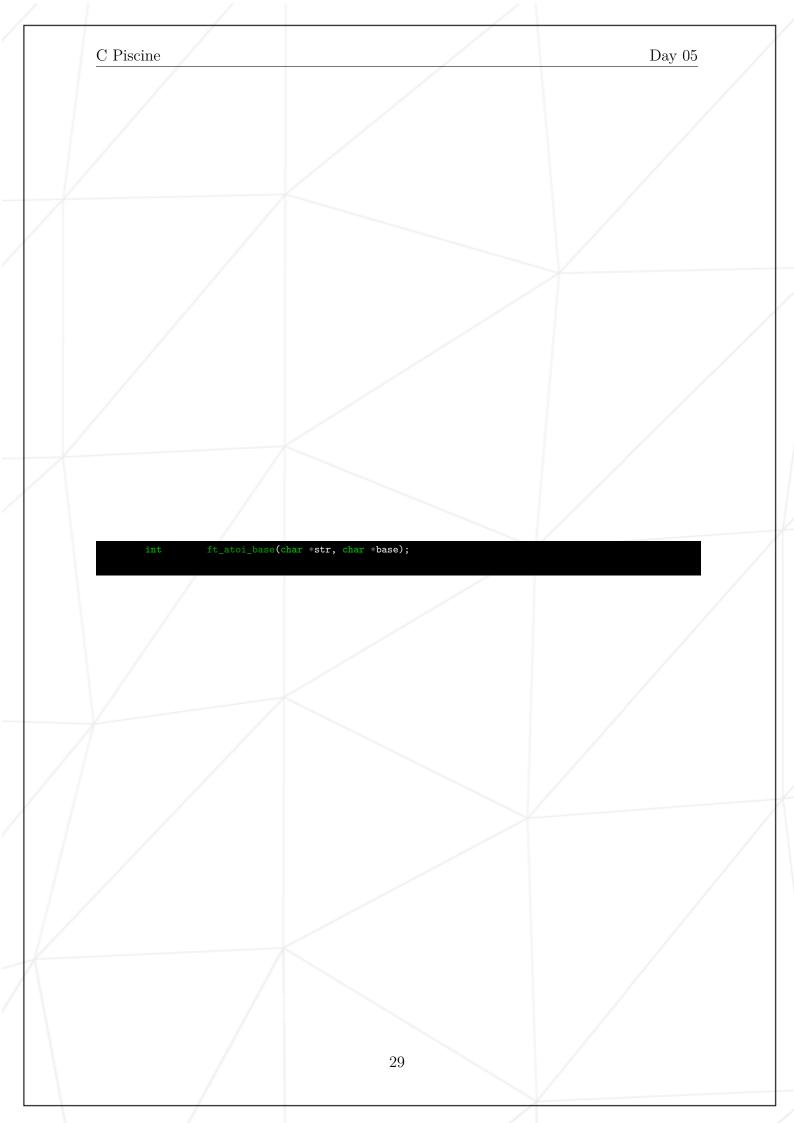
C Piscine Day 05 \circ etc. • Here's how it should be prototyped : ft_putnbr_base(int nbr, char *base); 27

Chapter XXIV

Exercise 21: ft_atoi_base

	Exercise 21	
/	ft_atoi_base	
Turn-in directory : $ex21/$		
Files to turn in: ft_atoi_b	ase.c	
Allowed functions : None		

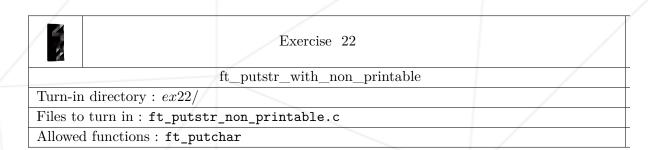
- Create a function that returns a number. This number is shaped as a string of characters.
- The string of characters reveals the number in a specific base, given as a second parameter.
- The function must handle negative numbers.
- The function must handle signs like man atoi.
- If there's an invalid argument, the function should return 0. Examples of invalid arguments :
 - str is an empty string;
 - the base is empty or size of 1;
 - str contains characters that aren't part of the base, or aren't + nor ;
 - the base contains the same character twice;
 - \circ the base contains + or -;
 - \circ etc.
- Here's how it should be prototyped:



Chapter XXV

Exercise 22:

ft_putstr_non_printable



- Create a function that displays a string of characters onscreen. If this string contains characters that aren't printable, they'll have to be displayed in the shape of hexadecimals (lowercase), preceded by a "backslash".
- For example :

Coucou\ntu vas bien ?

• The function should display :

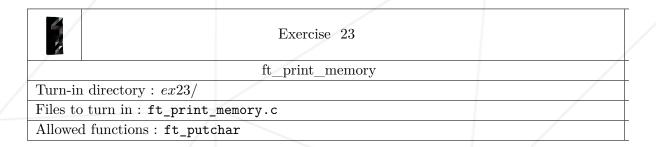
Coucou\Oatu vas bien ?

• Here's how it should be prototyped:

void ft_putstr_non_printable(char *str);

Chapter XXVI

Exercise 23: ft_print_memory



- Create a function that displays the memory area onscreen.
- The display of this memory area should be split into three columns :
 - The hexadecimal address of the first line's first character;
 - The content in hexadecimal;
 - The content in printable characters.
- If a character is non-printable, it'll be replaced by a dot.
- Each line should handle sixteen characters.
- If size equals to 0, nothing should be displayed.

C Piscine Day 05

• Example:

```
guilla_i@seattle $> ./ft_print_memory
00000000: 5361 6c75 7420 6c65 7320 616d 696e 6368 Salut les aminch
00000010: 6573 2063 2765 7374 2063 6f6f 6c20 7368 es c'est cool sh
00000020: 6f77 206d 656d 206f 6e20 6661 6974 2064 ow mem on fait d
00000030: 6520 7472 7563 2074 6572 7269 626c 6500 e truc terrible.
00000040: 2e00 0102 0304 0506 0708 090e 0f1b 7f .......
guilla_i@seattle $> ./ft_print_memory | cat -te
00000000: 5361 6c75 7420 6c65 7320 616d 696e 6368 Salut les aminch$
00000010: 6573 2063 2765 7374 2063 6f6f 6c20 7368 es c'est cool sh$
00000020: 6f77 206d 656d 206f 6e20 6661 6974 2064 ow mem on fait d$
00000030: 6520 7472 7563 2074 6572 7269 626c 6500 e truc terrible.$
00000040: 2e00 0102 0304 0506 0708 090e 0f1b 7f .............$
guilla_i@seattle $>
```

• Here's how it should be prototyped :

```
void *ft_print_memory(void *addr, unsigned int size);
```

• It should return addr.