

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_	$_{ m putstr}$	6
IV	Exercise 01 : ft_	putnbr	7
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	nutnbr base	26

	C Piscine	Day 05	
	XXIV Exercise 21 : ft_atoi_base	28	
	XXV Exercise 22 : ft_putstr_non_printable	30	
	XXVI Exercise 23 : ft_print_memory	31	
4			
1 /			
X			
	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.
- 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 -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. I'm so sorry. Uh, I mean like, what, we're going to bring kids into this world with that over their heads? 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 to be with someone who uses spaces over tabs.
- 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_p	utstr.c	
Allowed functions: ft	_putchar	
Notes : n/a		

- 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 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_putnbr.c		
Allowed functions: ft_putchar			
	Notes : n/a		

- 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:
  - $\circ$  ft\_putnbr(42) displays "42".

#### Chapter V

Exercise 02 : ft\_atoi

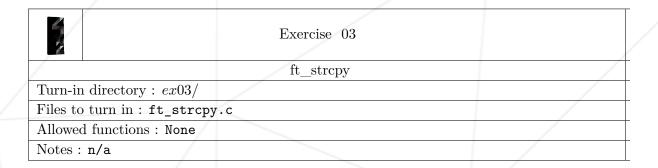
Exerc	ise 02
ft_	atoi
Turn-in directory : $ex02/$	
Files to turn in : ft_atoi.c	
Allowed functions : None	
Notes : n/a	

- 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).
  - $\bullet$  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

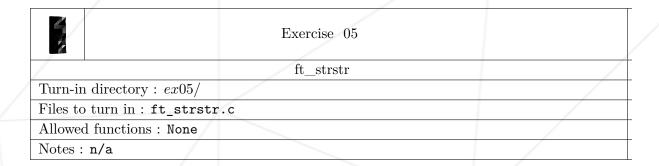
	Exercise 04	
/	ft_strncpy	
Turn-in directory : $ex04/$		
Files to turn in : ft_strncp	ру.с	
Allowed functions: None		
Notes : n/a		

- 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



- $\bullet$  Reproduce the behavior of the function  ${\tt strstr}$  (man  ${\tt strstr}).$
- Here's how it should be prototyped :

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

## Chapter IX

Exercise 06: ft\_strcmp

	Exercise 06	
/	ft_strcmp	
Turn-in directory : $ex06/$		
Files to turn in : ft_strcmp.c		
Allowed functions : None		
Notes : n/a		

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

nt ft\_strcmp(char \*s1, char \*s2);

## Chapter X

# Exercise 07: ft\_strncmp

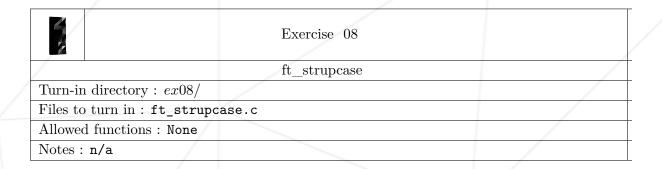
	Exercise 07	
/	ft_strncmp	
Turn-in directory : $ex07/$		
Files to turn in : ft_str	ncmp.c	
Allowed functions: None		
Notes : n/a		

- 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



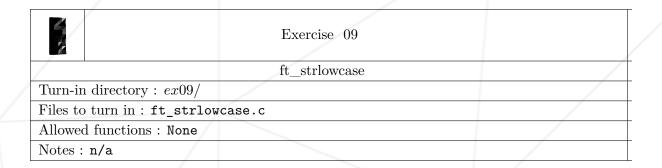
- Create a function that transforms every letter of every word to uppercase.
- Here's how it should be prototyped :

char \*ft\_strupcase(char \*str);

• It should return str.

#### Chapter XII

## Exercise 09: ft\_strlowcase



- $\bullet$  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

	Exercise 10	
/	ft_strcapitalize	
Turn-in directory : $ex10$	/	
Files to turn in : ft_str	capitalize.c	
Allowed functions: None		
Notes : n/a		

- 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:

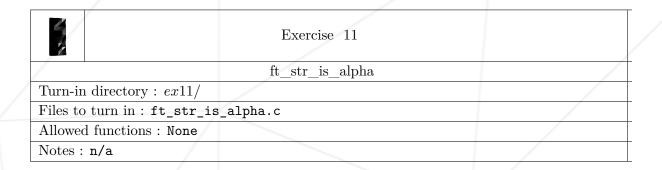
```
salut, comment tu vas ? 42mots quarante-deux; cinquante+et+un
```

• 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

	Exercise 12	
/	ft_str_is_numeric	
Turn-in directory : $ex12/$		
Files to turn in : ft_str_is	s_numeric.c	
Allowed functions : None		
Notes : n/a		

- 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

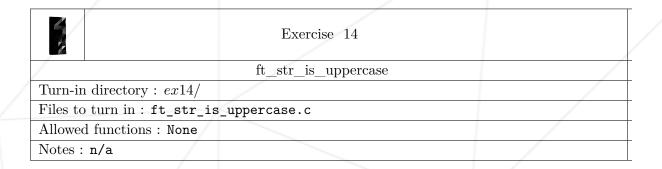
	Exercise 13	
/	ft_str_is_lowercase	
Turn-in directory : $ex13/$		
Files to turn in : ft_str_is_	lowercase.c	
Allowed functions : None		
Notes : n/a		

- 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

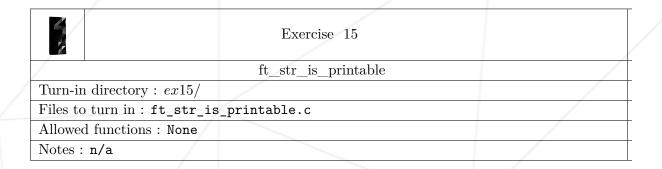


- Create a function that returns 1 if the string given as a parameter contains only uppercase alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

int ft\_str\_is\_uppercase(char \*str);

## Chapter XVIII

Exercise 15: ft\_str\_is\_printable

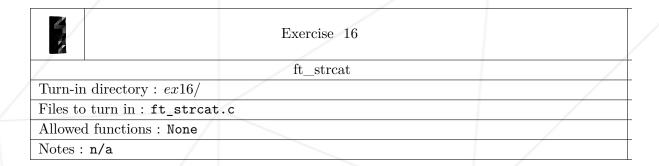


- Create a function that returns 1 if the string given as a parameter contains only printable characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

int ft\_str\_is\_printable(char \*str);

## Chapter XIX

Exercise 16: ft\_strcat

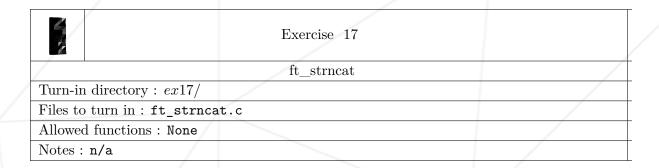


- $\bullet$  Reproduce the behavior of the function  $\mathtt{strcat}$  (man strcat).
- $\bullet$  Here's how it should be prototyped :

char \*ft\_strcat(char \*dest, char \*src);

## Chapter XX

# Exercise 17: ft\_strncat

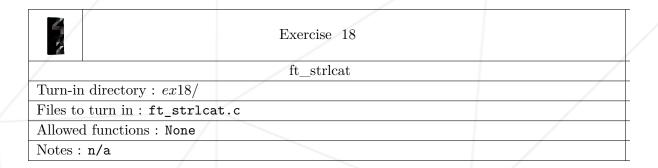


- Reproduce the behavior of the function strncat (man strncat).
- Here's how it should be prototyped :

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

	Exercise 19	
/	ft_strlcpy	
Turn-in directory : $ex19/$		
Files to turn in : ft_strlcpy	<i>y</i> .c	
Allowed functions : None		
Notes : n/a		

- 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_putchar	
Notes: n/a	

- 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;
  - $\circ$  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;

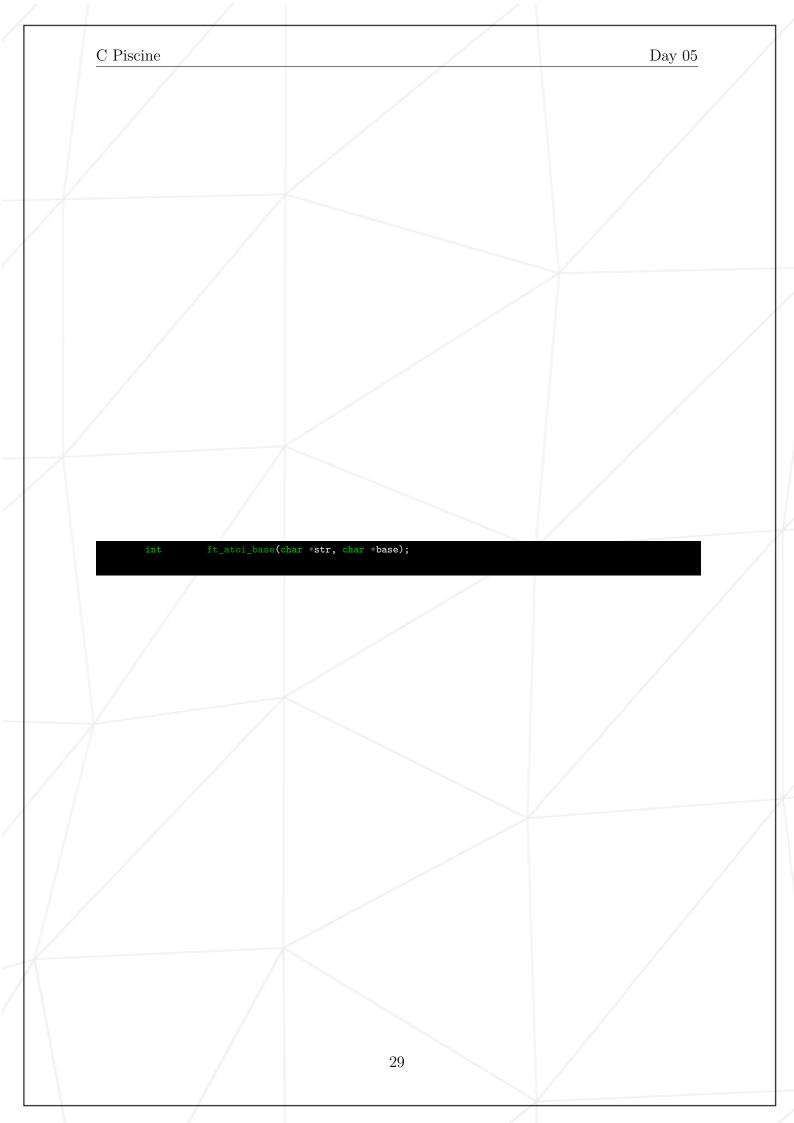
C Piscine Day 05  $\circ$  base contains + or -;  $\circ$  etc. • Here's how it should be prototyped : ft\_putnbr\_base(int nbr, char \*base); void

#### Chapter XXIV

#### Exercise 21: ft\_atoi\_base

	Exercise 21	
/	ft_atoi_base	
Turn-in directory : $ex21/$		
Files to turn in: ft_ato:	i_base.c	
Allowed functions: None		
Notes : n/a		

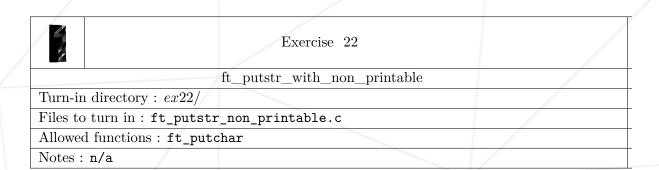
- Create a function that returns a number. This number is shaped as a string of characters.
- The string of characters reveals the number in a specific base, given as a second parameter.
- The function must handle negative numbers.
- The function must handle signs like man atoi.
- If there's an invalid 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 -;
  - o 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:

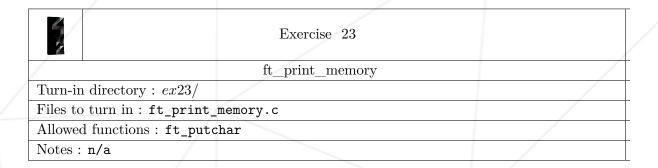
Coucou\Oatu vas bien ?

• 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 $> ./ft_print_memory | cat -te
00000000: 5361 6c75 7420 6c65 7320 616d 696e 6368 Salut les aminch$
0000010: 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.