

# C Bootcamp

Day 05

 $Staff\ We Think Code\_\ {\tt info@wethinkcode.co.za}$ 

Summary: This document is the subject for Day05 of the C Bootcamp @ WeThinkCode.

## Contents

1	Instructions	2
II	Foreword	4
III	Exercise 00 : ft_strcpy	5
IV	Exercise 01 : ft_strncpy	6
$\mathbf{V}$	Exercise 02 : ft_strstr	7
VI	Exercise 03 : ft_strcmp	8
VII	Exercise 04 : ft_strncmp	9
VIII	Exercise 05 : ft_strupcase	10
IX	Exercise 06 : ft_strlowcase	
$\mathbf{X}$	Exercise 07 : ft_strcapitalize	12
XI	Exercise 08 : ft_str_is_alpha	13
XII	Exercise 09 : ft_str_is_numeric	14
XIII	Exercise 10 : ft_str_is_lowercase	15
XIV	Exercise 11 : ft_str_is_uppercase	16
XV	Exercise 12 : ft_str_is_printable	17
XVI	Exercise 13 : ft_streat	18
XVII	Exercise 14 : ft_strncat	19
XVIII	Exercise 15 : ft_strlcat	20
XIX	Exercise 16 : ft_strlcpy	21
XX	Exercise 17 : ft_putnbr_base	22
XXI	Exercise 18 : ft_atoi_base	24
XXII	Exercise 19 : ft_putstr_non_printable	26
XXIII	Exercise 20: ft. print. memory	27

#### 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.

C Bootcamp

Day 05

- 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 Bootcamp" 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

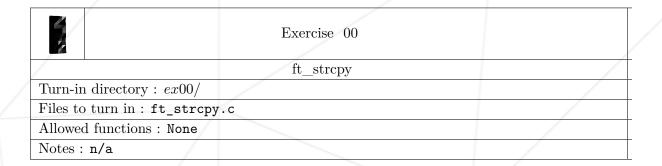
Here's a quote from Bessie Anderson Stanley which i liked and I think you will as well.

"He has achieved success who has lived well, laughed often, and loved much; Who has enjoyed the trust of pure women, the respect of intelligent men and the love of little children; Who has filled his niche and accomplished his task; Who has never lacked appreciation of Earth's beauty or failed to express it; Who has left the world better than he found it, Whether an improved poppy, a perfect poem, or a rescued soul; Who has always looked for the best in others and given them the best he had; Whose life was an inspiration; Whose memory a benediction."

If you don't well that's not supposed to affect your work for this day.

#### Chapter III

Exercise 00: ft\_strcpy

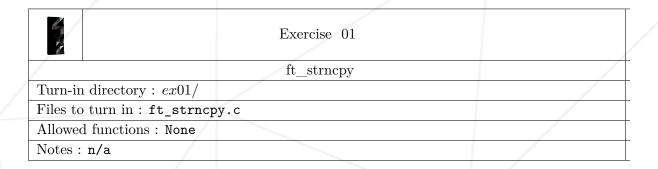


- Reproduce the behavior of the function strcpy (man strcpy).
- Here's how it should be prototyped :

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

#### Chapter IV

## Exercise 01: 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 V

## Exercise 02 : ft\_strstr

	Exercise 02	
	ft_strstr	
Turn-in directory : $ex02$	/	
Files to turn in : ft_str	str.c	
Allowed functions: None		
Notes : n/a		

- 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 VI

## Exercise 03: ft\_strcmp

	Exercise 03	
/	ft_strcmp	
Turn-in directory : $ex03/$		
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 VII

# Exercise 04 : ft\_strncmp

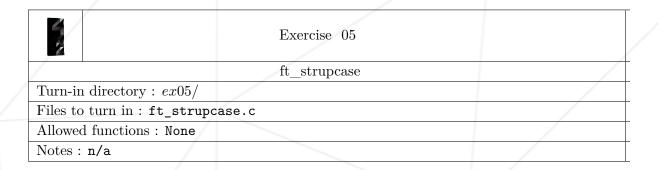
	Exercise 04	
	ft_strncmp	
Turn-in directory : $ex04/$		
Files to turn in: ft_strncm	p.c	
Allowed functions : None		
Notes : n/a		

- 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 VIII

Exercise 05: ft\_strupcase



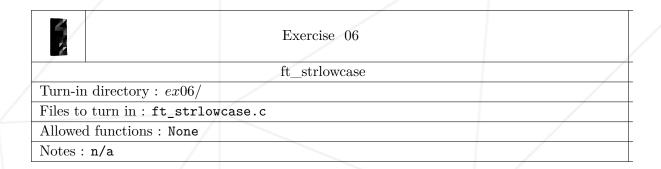
- Create a function that transforms every letter of every word to uppercase.
- Here's how it should be prototyped :

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

• It should return str.

#### Chapter IX

Exercise 06: ft\_strlowcase



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

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

• It should return str.

#### Chapter X

#### Exercise 07: ft\_strcapitalize

Exercise 07	
ft_strcapitalize	
Turn-in directory: $ex07/$	
Files to turn in : ft_strcapitalize.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:

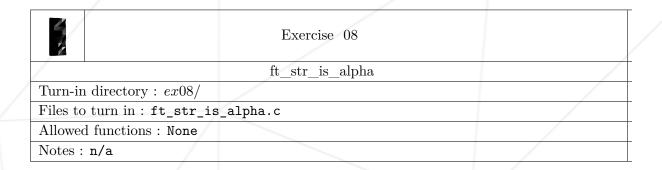
```
salut, comment tu vas ? 42mots quarante-deux; cinquante+et+un
```

• Becomes:

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

#### Chapter XI

Exercise 08: ft\_str\_is\_alpha

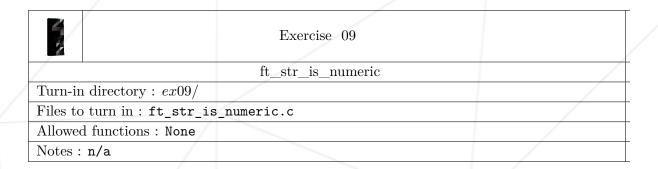


- Create a function that returns 1 if the string given as a parameter contains only alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

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

#### Chapter XII

Exercise 09: ft\_str\_is\_numeric

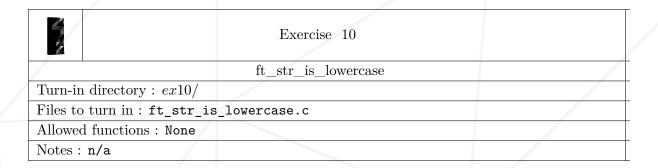


- Create a function that returns 1 if the string given as a parameter contains only digits, and 0 if it contains any other character.
- Here's how it should be prototyped :

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

#### Chapter XIII

Exercise 10: ft\_str\_is\_lowercase

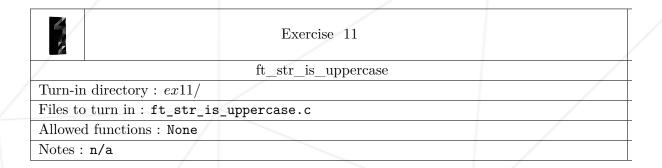


- Create a function that returns 1 if the string given as a parameter contains only lowercase alphabetical characters, and 0 if it contains any other character.
- Here's how it should be prototyped :

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

#### Chapter XIV

## Exercise 11 : 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 XV

## Exercise 12: ft\_str\_is\_printable

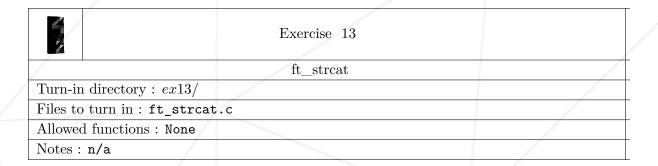
	Exercise 12	
	ft_str_is_printable	
Turn-in directory : $ex12/$		
Files to turn in : ft_str_is	_printable.c	
Allowed functions: None		
Notes : n/a		

- 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 XVI

Exercise 13: ft\_strcat

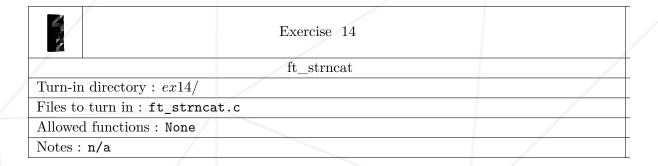


- $\bullet$  Reproduce the behavior of the function  ${\tt strcat}$  (man  ${\tt strcat}).$
- $\bullet$  Here's how it should be prototyped :

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

## Chapter XVII

# Exercise 14: ft\_strncat

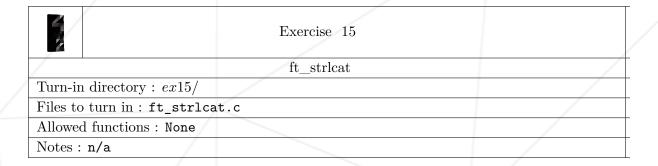


- Reproduce the behavior of the function strncat (man strncat).
- Here's how it should be prototyped :

char \*ft\_strncat(char \*dest, char \*src, int nb);

## Chapter XVIII

Exercise 15: 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 XIX

Exercise 16: ft\_strlcpy

	Exercise 16	
/	ft_strlcpy	
Turn-in directory : $ex16/$		
Files to turn in : ft_strlcp	ру.с	
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 XX

#### Exercise 17: ft\_putnbr\_base

	Exercise 17	
/	ft_putnbr_base	
Turn-in directory : $ex17$	7/	
Files to turn in : ft_pu	tnbr_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 Bootcamp

Day 05

- $\circ$  base contains + or -;
- $\circ$  etc.
- Here's how it should be prototyped :

void ft\_putnbr\_base(int nbr, char \*base);

#### Chapter XXI

#### Exercise 18: ft\_atoi\_base

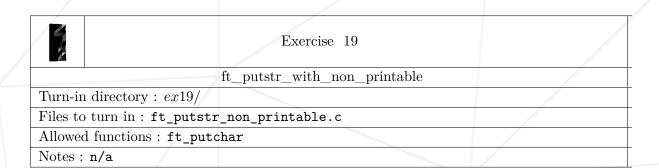
	Exercise 18	
/	ft_atoi_base	
Turn-in directory : $ex18/$		
Files to turn in : ft_atoi_h	base.c	
Allowed functions : None		
Notes : n/a		

- 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 :
  - $\circ$  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:

C Bootcamp	Day 05
int ft_atoi_bas	se(char *str, char *base);
	or.
	25

#### Chapter XXII

# Exercise 19: 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 XXIII

#### Exercise 20: ft\_print\_memory

	Exercise 20	
/	ft_print_memory	
Turn-in directory : $ex2$	0/	
Files to turn in : ft_pr	rint_memory.c	
Allowed functions: ft	putchar	
Notes : n/a		

- 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 Bootcamp 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.