



C Piscine

C 02

*Summary:* このドキュメントは、C Piscine @ 42の C 02モジュール用の課題である。

# Contents

I	Instructions	2
II	Foreword	4
III	Exercise 00 : ft_strcpy	5
IV	Exercise 01 : ft_strncpy	6
V	Exercise 02 : ft_str_is_alpha	7
VI	Exercise 03 : ft_str_is_numeric	8
VII	Exercise 04 : ft_str_is_lowercase	9
VIII	Exercise 05 : ft_str_is_uppercase	10
IX	Exercise 06 : ft_str_is_printable	11
X	Exercise 07 : ft_strupcase	12
XI	Exercise 08 : ft_strlowcase	13
XII	Exercise 09 : ft_strcapitalize	14
XIII	Exercise 10 : ft_strlcpy	15
XIV	Exercise 11 : ft_putstr_non_printable	16
XV	Exercise 12 : ft_print_memory	17
XVI	Submission and peer-evaluation	19

# Chapter I

## Instructions

- 課題に関する噂に惑わされないよう気をつけ、信用しないこと。
- この書類は、提出前に変更になる可能性があるため、気をつけること。
- ファイルとディレクトリへの権限があることを、あらかじめ確認すること。
- すべての課題は、提出手順に従い行うこと。
- 課題の確認と評価は、あなたの周りにいるPiscine受験者により行われる。
- 課題の確認と評価は、Piscine受験者に加えて、Moulinetteと呼ばれるプログラムによっても行われる。
- Moulinetteは、大変細かい評価を行う。これはすべて自動で行われるため、交渉の余地はない。
- Moulinetteは、コーディング規範（Norm）を遵守しないコードを解釈することができない。そのため、Moulinetteはnorminetteと呼ばれるプログラムを使用し、あなたのファイルがコーディング規範を遵守しているか確認を行う。せっかくの取り組みが、norminetteの確認により無駄にならないよう、気をつけること。
- 問題は、簡単なものから徐々に難しくなるように並べられている。簡単な問題が解けていない場合は、難しい問題が解けていたとしても 加点されることはない。
- 使用が禁止されている関数を使用した場合は、不正とみなされる。不正者は-42の評価をつけられ、この評価に対する交渉の余地はない。
- 課題がプログラムの提出を要求する場合のみ、main()関数を提出すること。
- Moulinetteは以下のフラッグを用いて、ccでコンパイルする。 -Wall -Wextra -Werror
- プログラムがコンパイルされなかった場合、評価は0になる。
- 課題で指定されていないものは、どんなファイルもディレクトリ内に置かないこと。

- 質問がある場合は、隣の人に聞くこと。それでも分からない場合は、反対側の席の人に聞くこと。
- 助けてくれるのは、Google / 人間 / インターネット / ...と呼ばれているものたちである。
- 出力例には、問題文に明記されていない細部まで表示されている場合があるため、入念に確認すること。



norminetteは、 `-R CheckForbiddenSourceHeader` をオプションに追加しなければならない。Moulinetteも、このオプションを使用する。

# 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\_strcpy


	Exercise 00
	ft_strcpy
	提出するディレクトリ : <i>ex00/</i>
	提出するファイル : <i>ft_strcpy.c</i>
	使用可能な関数 : None

- strcpy関数の動作を再現しなさい。(man strcpy)
- プロトタイプ例)

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

# Chapter IV

## Exercise 01 : ft\_strncpy


	Exercise 01
	ft_strncpy
	提出するディレクトリ : <i>ex01/</i>
	提出するファイル : <i>ft_strncpy.c</i>
	使用可能な関数 : None

- strncpy関数の動作を再現しなさい。(man strncpy)
- プロトタイプ例)

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

# Chapter V

## Exercise 02 : ft\_str\_is\_alpha

	Exercise 02
	ft_str_is_alpha
提出するディレクトリ : <i>ex02/</i>	
提出するファイル : <b>ft_str_is_alpha.c</b>	
使用可能な関数 : None	

- パラメータとして与えられた文字列に、英字のみが含まれている場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成せよ。
- プロトタイプ例)


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

- `str`が空の場合、関数が1を返すように実装すること。



# Chapter VI

## Exercise 03 : ft\_str\_is\_numeric

	Exercise 03
ft_str_is_numeric	
提出するディレクトリ : ex03/	
提出するファイル : ft_str_is_numeric.c	
使用可能な関数 : None	


- パラメータとして与えられた文字列に、数字のみが含まれている場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成せよ。
- プロトタイプ例)

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

- strが空の場合、関数が1を返すように実装すること。

# Chapter VII

## Exercise 04 : ft\_str\_is\_lowercase

	Exercise 04
ft_str_is_lowercase	
提出するディレクトリ : ex04/	
提出するファイル : ft_str_is_lowercase.c	
使用可能な関数 : None	


- パラメータとして与えられた文字列に、小文字のアルファベットのみが含まれている場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成せよ。
- プロトタイプ例)

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

- strが空の場合、関数が1を返すように実装すること。

# Chapter VIII

## Exercise 05 : ft\_str\_is\_uppercase

	Exercise 05
ft_str_is_uppercase	
提出するディレクトリ : <i>ex05/</i>	
提出するファイル : <i>ft_str_is_uppercase.c</i>	
使用可能な関数 : None	


- パラメータとして与えられた文字列に、大文字のアルファベットのみが含まれる場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成せよ。
- プロトタイプ例)

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

- *str*が空の場合、関数が1を返すように実装すること。

## Chapter IX

### Exercise 06 : ft\_str\_is\_printable

	Exercise 06
	ft_str_is_printable
	提出するディレクトリ : ex06/
	提出するファイル : ft_str_is_printable.c
	使用可能な関数 : None


- パラメータとして与えられた文字列に、表示文字のみが含まれている場合は1を返し、それ以外の文字が含まれている場合は0を返す関数を作成せよ。
- プロトタイプ例)

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

- strが空の場合、関数が1を返すように実装すること。

# Chapter X

## Exercise 07 : ft\_strupcase

	Exercise 07
	ft_strupcase
	提出するディレクトリ : <i>ex07/</i>
	提出するファイル : <b>ft_strupcase.c</b>
	使用可能な関数 : None


- 文字列にあるすべての文字を、大文字に変換する関数を作成せよ。
- プロトタイプ例)

```
char *ft_strupcase(char *str);
```

- 関数がstr を返すように実装すること。

# Chapter XI

## Exercise 08 : ft\_strlowcase

	Exercise 08
	ft_strlowcase
	提出するディレクトリ : <i>ex08/</i>
	提出するファイル : <i>ft_strlowcase.c</i>
	使用可能な関数 : None


- 文字列にあるすべての文字を、小文字に変換する関数を作成せよ。
- プロトタイプ例)

```
char *ft_strlowcase(char *str);
```

- 関数がstr を返すように実装すること。

# Chapter XII

## Exercise 09 : ft\_strcapitalize

	Exercise 09
	ft_strcapitalize
	提出するディレクトリ : <i>ex09/</i>
	提出するファイル : <i>ft_strcapitalize.c</i>
	使用可能な関数 : None

- 各単語の最初の文字を大文字に変換し、それ以外の文字を小文字に変換する関数を作成せよ。
- 単語とは英数字の文字列である。
- プロトタイプ例)

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

- 関数が、strを返すように実装すること。
- 例)


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

- 上記の入力に対する出力は、以下のようになる。

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

# Chapter XIII

## Exercise 10 : ft\_strlcpy

	Exercise 10
	ft_strlcpy
	提出するディレクトリ : <i>ex10/</i>
	提出するファイル : <i>ft_strlcpy.c</i>
	使用可能な関数 : None


- strlcpy関数の動作を再現しなさい。(man strlcpy)
- プロトタイプ例)

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



# Chapter XIV

## Exercise 11 : ft\_putstr\_non\_printable

	Exercise 11
ft_putstr_with_non_printable	
提出するディレクトリ : <i>ex11/</i>	
提出するファイル : <i>ft_putstr_non_printable.c</i>	
使用可能な関数 : <i>write</i>	

- 与えられた文字列を標準出力に出力する関数を作成せよ。文字列に出力不可能な文字が含まれている場合は、16進数（小文字）にして、その文字の前にバックスラッシュ “\” をつけて出力すること。

- 例)

```
Coucou\ntu vas bien ?
```

- 出力例)


```
Coucou\0atu vas bien ?
```

- プロトタイプ例)

```
void      ft_putstr_non_printable(char *str);
```

# Chapter XV

## Exercise 12 : ft\_print\_memory

	Exercise 12
	ft_print_memory
	提出するディレクトリ : <i>ex12/</i>
	提出するファイル : <i>ft_print_memory.c</i>
	使用可能な関数 : <i>write</i>

- メモリ領域を標準出力に出力する関数を作成せよ。
- メモリ領域は、以下のように、3つの列に分けて出力すること。
  - 行の最初の文字のアドレス（16進数で表したアドレス）と、 ":" が出力されていること。
  - 以下の例を参照し、2文字ごとにスペースを含む16進数に、必要に応じて追加のスペースを入れること。
  - アドレスにある要素を、表示文字に変換すること。
- 文字が表示文字ではない場合は、ドット "." に置き換えられる。
- 各行は、16文字で構成されていること。
- サイズが0の場合は、何も出力しないこと。

- 例)

```
$> ./ft_print_memory
000000010a161f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin
000000010a161f50: 6368 6573 090a 0963 0720 6573 7420 666f ches...c. est fo
000000010a161f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on
000000010a161f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.
000000010a161f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..
000000010a161f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000 ..lol.lol. .
$> ./ft_print_memory | cat -te
0000000107ff9f40: 426f 6e6a 6f75 7220 6c65 7320 616d 696e Bonjour les amin$
0000000107ff9f50: 6368 6573 090a 0963 0720 6573 7420 666f ches...c. est fo$
0000000107ff9f60: 7509 746f 7574 0963 6520 7175 206f 6e20 u.tout.ce qu on $
0000000107ff9f70: 7065 7574 2066 6169 7265 2061 7665 6309 peut faire avec.$
0000000107ff9f80: 0a09 7072 696e 745f 6d65 6d6f 7279 0a0a ..print_memory..$
0000000107ff9f90: 0a09 6c6f 6c2e 6c6f 6c0a 2000 ..lol.lol. .$.
$>
```

- プロトタイプ例)

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

- 関数が、addr を返すように実装すること。

# Chapter XVI

## Submission and peer-evaluation

課題は、いつも通り Git リポジトリに提出すること。リポジトリ内の提出物のみが、レビュー中の評価対象となる。ファイルの名前が正しいことを確認すること。



この課題の要件で求められているファイルのみを提出すること。