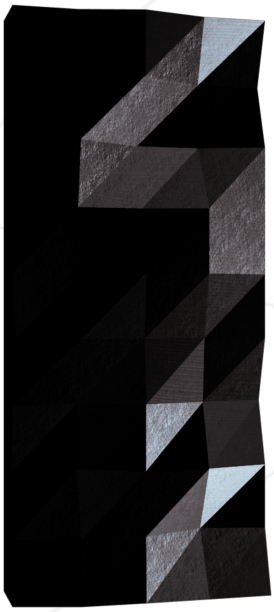

Piscine C

Jour 05

Staff 42 piscine@42.fr

Résumé: Ce document est le sujet du jour 05 de la piscine C de 42.

Table des matières

Chapitre I

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices.
- Vos exercices seront corrigés par vos camarades de piscine.
- En plus de vos camarades, vous serez corrigés par un programme appelé la Moulinette.
- La Moulinette est très stricte dans sa notation. Elle est totalement automatisée. Il est impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les surprises.
- La Moulinette n'est pas très ouverte d'esprit. Elle ne cherche pas à comprendre le code qui ne respecte pas la Norme. La Moulinette utilise le programme `norminette` pour vérifier la norme de vos fichiers. Comprendre par là qu'il est stupide de rendre un code qui ne passe pas la `norminette`.
- L'utilisation d'une fonction interdite est un cas de triche. Toute triche est sanctionnée par la note de `-42`.
- Si ft_putchar() est une fonction autorisée, nous compilerons avec notre `ft_putchar.c`.
- Vous ne devrez rendre une fonction main() que si nous vous demandons un programme.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- La Moulinette compile avec les flags -Wall -Wextra -Werror, et utilise `gcc`.
- Si votre programme ne compile pas, vous aurez `0`.
- Vous ne devez laisser dans votre répertoire aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices.

- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Votre manuel de référence s'appelle `Google / man / Internet / ...`.
- Pensez à discuter sur le forum Piscine de votre Intra !
- Lisez attentivement les exemples. Ils pourraient bien requérir des choses qui ne sont pas autrement précisées dans le sujet...
- Réfléchissez. Par pitié, par Odin ! Nom d'une pipe.

Pour cette journée, la norminette doit être lancée avec le flag `-R CheckForbiddenSourceHeader`. La moulinette l'utilisera aussi.

Chapitre II

Préambule

Voici une discussion extraite de la série `Silicon Valley` :

```
- 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.
```

Heureusement, vous n'êtes pas obliger d'utiliser emacs et votre barre espace pour compléter les exercices suivants.

Chapitre III

Exercice 00 : ft_putstr

	Exercice : 00
ft_putstr	
Dossier de rendu : $ex00/$	
Fichiers à rendre : `ft_putstr.c`	
Fonctions Autorisées : `ft_putchar`	

42 - Classics : Ces exercices sont incontournables et ne rapportent aucun points, mais il est imperatif de les valider pour accéder aux véritables exercices du jour.

- Écrire une fonction qui affiche un à un les caractères d'une chaîne à l'écran.
- L'adresse du premier caractère de la chaîne est contenue dans le pointeur passé en paramètre à la fonction.
- Elle devra être prototypée de la façon suivante :

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

Chapitre IV

Exercice 01 : ft_putnbr

	Exercice : 01
ft_putnbr	
Dossier de rendu : *ex*01/	
Fichiers à rendre : `ft_putnbr.c`	
Fonctions Autorisées : `ft_putchar`	

42 - Classics : Ces exercices sont incontournables et ne rapportent aucun points, mais il est imperatif de les valider pour accéder aux véritables exercices du jour.

- Écrire une fonction qui affiche un nombre passé en paramètre. La fonction devra être capable d'afficher la totalité des valeurs possibles dans une variable de type `int`.
- Elle devra être prototypée de la façon suivante :

```
void ft_putnbr(int nb);
```

- Par exemple :
  - `ft_putnbr(42)` affiche "42".

Chapitre V

Exercice 02 : ft_atoi

	Exercice : 02
ft_atoi	
Dossier de rendu : *ex*02/	
Fichiers à rendre : `ft_atoi.c`	
Fonctions Autorisées : `Aucune`	

42 - Classics : Ces exercices sont incontournables et ne rapportent aucun points, mais il est imperatif de les valider pour accéder aux véritables exercices du jour.

- Reproduire à l'identique le fonctionnement de la fonction `atoi` (man atoi).
- Elle devra être prototypée de la façon suivante :

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

Chapitre VI

Exercice 03 : ft_strcpy

	Exercice : 03
ft_strcpy	
Dossier de rendu : $ex03/$	
Fichiers à rendre : `ft_strcpy.c`	
Fonctions Autorisées : `Aucune`	

- Reproduire à l'identique le fonctionnement de la fonction `strcpy` (man strcpy).
- Elle devra être prototypée de la façon suivante :

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

Chapitre VII

Exercice 04 : ft_strncpy

	Exercice : 04
ft_strncpy	
Dossier de rendu : *ex*04/	
Fichiers à rendre : `ft_strncpy.c`	
Fonctions Autorisées : `Aucune`	

- Reproduire à l'identique le fonctionnement de la fonction `strncpy` (man strncpy).
- Elle devra être prototypée de la façon suivante :

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

Chapitre VIII

Exercice 05 : ft_strstr

	Exercice : 05
ft_strstr	
Dossier de rendu : *ex*05/	
Fichiers à rendre : `ft_strstr.c`	
Fonctions Autorisées : `Aucune`	

- Reproduire à l'identique le fonctionnement de la fonction `strstr` (man strstr).
- Elle devra être prototypée de la façon suivante :

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

Chapitre IX

Exercice 06 : ft_strcmp

	Exercice : 06
ft_strcmp	
Dossier de rendu : $ex06/$	
Fichiers à rendre : `ft_strcmp.c`	
Fonctions Autorisées : `Aucune`	

- Reproduire à l'identique le fonctionnement de la fonction `strcmp` (man strcmp).
- Elle devra être prototypée de la façon suivante :

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

Chapitre X

Exercice 07 : ft_strncmp

	Exercice : 07
ft_strncmp	
Dossier de rendu : $ex07/$	
Fichiers à rendre : `ft_strncmp.c`	
Fonctions Autorisées : `Aucune`	

- Reproduire à l'identique le fonctionnement de la fonction `strncmp` (man strncmp).
- Elle devra être prototypée de la façon suivante :

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

Chapitre XI

Exercice 08 : ft_strupcase

	Exercice : 08
ft_strupcase	
Dossier de rendu : $ex08/$	
Fichiers à rendre : `ft_strupcase.c`	
Fonctions Autorisées : `Aucune`	

- Écrire une fonction qui met en majuscule chaque lettre de chaque mot.
- Elle devra être prototypée de la façon suivante :

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

- Elle devra renvoyer `str`.

Chapitre XII

Exercice 09 : ft_strlowcase

	Exercice : 09
ft_strlowcase	
Dossier de rendu : *ex*09/	
Fichiers à rendre : `ft_strlowcase.c`	
Fonctions Autorisées : `Aucune`	

- Écrire une fonction qui met en minuscule chaque lettre de chaque mot.
- Elle devra être prototypée de la façon suivante :

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

- Elle devra renvoyer `str`.

Chapitre XIII

Exercice 10 : ft_strcapitalize

	Exercice : 10
ft_strcapitalize	
Dossier de rendu : $ex10/$	
Fichiers à rendre : `ft_strcapitalize.c`	
Fonctions Autorisées : `Aucune`	

- Écrire une fonction qui met en majuscule la première lettre de chaque mot et le reste du mot en minuscule.
- Un mot est une suite de caractères alphanumériques.
- Elle devra être prototypée de la façon suivante :

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

- Elle devra renvoyer `str`.
- Par exemple :

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

- Doit donner :

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

Chapitre XIV

Exercice 11 : ft_str_is_alpha

	Exercice : 11
ft_str_is_alpha	
Dossier de rendu : $ex11/$	
Fichiers à rendre : `ft_str_is_alpha.c`	
Fonctions Autorisées : `Aucune`	

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des caractères alphabétiques et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

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

- Elle devra renvoyer 1 si `str` est une chaîne vide.

Chapitre XV

Exercice 12 : ft_str_is_numeric

	Exercice : 12
ft_str_is_numeric	
Dossier de rendu : *ex*12/	
Fichiers à rendre : `ft_str_is_numeric.c`	
Fonctions Autorisées : `Aucune`	

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des chiffres et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

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

- Elle devra renvoyer 1 si `str` est une chaîne vide.

Chapitre XVI

Exercice 13 : ft_str_is_lowercase

	Exercice : 13
ft_str_is_lowercase	
Dossier de rendu : *ex*13/	
Fichiers à rendre : `ft_str_is_lowercase.c`	
Fonctions Autorisées : `Aucune`	

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des caractères alphabétiques minuscules et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

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

- Elle devra renvoyer 1 si `str` est une chaîne vide.

Chapitre XVII

Exercice 14 : ft_str_is_uppercase

	Exercice : 14
ft_str_is_uppercase	
Dossier de rendu : $ex14/$	
Fichiers à rendre : `ft_str_is_uppercase.c`	
Fonctions Autorisées : `Aucune`	

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des caractères alphabétiques majuscules et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

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

- Elle devra renvoyer 1 si `str` est une chaîne vide.

Chapitre XVIII

Exercice 15 : ft_str_is_printable

	Exercice : 15
ft_str_is_printable	
Dossier de rendu : $ex15/$	
Fichiers à rendre : `ft_str_is_printable.c`	
Fonctions Autorisées : `Aucune`	

- Écrire une fonction qui renvoie 1 si la chaîne passée en paramètre ne contient que des caractères affichables et renvoie 0 si la fonction contient d'autres types de caractères.
- Elle devra être prototypée de la façon suivante :

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

- Elle devra renvoyer 1 si `str` est une chaîne vide.

Chapitre XIX

Exercice 16 : ft_strcat

	Exercice : 16
ft_strcat	
Dossier de rendu : $ex16/$	
Fichiers à rendre : `ft_strcat.c`	
Fonctions Autorisées : `Aucune`	

- Reproduire à l'identique le fonctionnement de la fonction `strcat` (man strcat).
- Elle devra être prototypée de la façon suivante :

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

Chapitre XX

Exercice 17 : ft_strncat

	Exercice : 17
ft_strncat	
Dossier de rendu : $ex17/$	
Fichiers à rendre : `ft_strncat.c`	
Fonctions Autorisées : `Aucune`	

- Reproduire à l'identique le fonctionnement de la fonction `strncat` (man strncat).
- Elle devra être prototypée de la façon suivante :

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

Chapitre XXI

Exercice 18 : ft_strlcat

	Exercice : 18
ft_strlcat	
Dossier de rendu : *ex*18/	
Fichiers à rendre : `ft_strlcat.c`	
Fonctions Autorisées : `Aucune`	

- Reproduire à l'identique le fonctionnement de la fonction `strlcat` (man strlcat).
- Elle devra être prototypée de la façon suivante :

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

Chapitre XXII

Exercice 19 : ft_strlcpy

	Exercice : 19
ft_strlcpy	
Dossier de rendu : $ex19/$	
Fichiers à rendre : `ft_strlcpy.c`	
Fonctions Autorisées : `Aucune`	

- Reproduire à l'identique le fonctionnement de la fonction `strlcpy` (man strlcpy).
- Elle devra être prototypée de la façon suivante :

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

Chapitre XXIII

Exercice 20 : ft_putnbr_base

	Exercice : 20
ft_putnbr_base	
Dossier de rendu : $ex20/$	
Fichiers à rendre : `ft_putnbr_base.c`	
Fonctions Autorisées : `ft_putchar`	

- Écrire une fonction qui affiche un nombre à l'écran dans une base donnée.
- Ce nombre est fourni sous la forme d'un `int` et la base sous la forme d'une `chaîne de caractères`.
- La base contient tous les symboles utilisables pour afficher le nombre :
  - `0123456789` est la base couramment utilisée pour représenter nos nombres décimaux ;
  - `01` est une base binaire ;
  - `0123456789ABCDEF` est une base hexadecimale ;
  - `poneyvif` est une base octale.
- La fonction doit gérer les nombres négatifs.
- Si un paramètre contient une erreur la fonction n'affiche rien. Une erreur peut être :
  - base est vide ou est de taille 1 ;
  - base contient deux fois le même caractère ;
  - base contient les caractères + ou - ;
  - etc.
- Elle devra être prototypée de la façon suivante :

```
void        ft_putnbr_base(int nbr, char *base);
```

Chapitre XXIV

Exercice 21 : ft_atoi_base

	Exercice : 21
ft_atoi_base	
Dossier de rendu : *ex*21/	
Fichiers à rendre : `ft_atoi_base.c`	
Fonctions Autorisées : `Aucune`	

- Écrire une fonction qui renvoie un nombre. Ce nombre est connu sous la forme d'une `chaîne de caractères`.
- La chaîne de caractères exprime le nombre dans une base particulière passée en second paramètre.
- La fonction doit gérer les nombres négatifs.
- La fonction doit gérer les signes comme `man atoi`.
- Si un paramètre contient une erreur la fonction renvoie 0. Une erreur peut être :
  - str est une chaîne vide ;
  - la base est vide ou est de taille 1 ;
  - str contient des caractères qui ne sont pas dans la base et ne sont pas + ou - ;
  - la base contient deux fois le même caractère ;
  - la base contient les caractères + ou - ;
  - etc.
- Elle devra être prototypée de la façon suivante :

```
int        ft_atoi_base(char *str, char *base);
```

Chapitre XXV

Exercice 22 : ft_putstr_non_printable

	Exercice : 22
ft_putstr_with_non_printable	
Dossier de rendu : *ex*22/	
Fichiers à rendre : `ft_putstr_non_printable.c`	
Fonctions Autorisées : `ft_putchar`	

- Écrire une fonction qui affiche une chaîne de caractères à l'écran. Si cette chaîne contient des caractères non-imprimables, ils devront être affichés sous forme hexadécimale (en minuscules) en les précédant d'un "backslash".
- Par exemple, avec ce paramètre :

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

- La fonction devra afficher :

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

- Elle devra être prototypée de la façon suivante :

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

Chapitre XXVI

Exercice 23 : ft_print_memory

	Exercice : 23
ft_print_memory	
Dossier de rendu : $ex23/$	
Fichiers à rendre : `ft_print_memory.c`	
Fonctions Autorisées : `ft_putchar`	

- Écrire une fonction qui affiche une zone mémoire à l'écran.
- L'affichage de la zone mémoire est séparée en trois colonnes :
  - L'adresse en hexadécimal du premier caractère de la ligne ;
  - Le contenu en hexadécimal ;
  - Le contenu en caractères imprimables.
- Si un caractère est non-imprimable il sera remplacé par un point.
- Chaque ligne doit gérer seize caractères.
- Si `size` est égale à 0, rien ne sera affiché.

- Exemple :

```
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 $>
```

- Elle devra être prototypée de la façon suivante :

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

- Elle devra renvoyer `addr`.