



Piscine C

Jour 12

Staff 42 piscine@42.fr

Résumé: Ce document est le sujet du jour 12 de la piscine C de 42.

Table des matières

| | | |
|------------|-----------------------------------|----------|
| I | Consignes | 2 |
| II | Préambule | 4 |
| III | Exercice 00 : display_file | 5 |
| IV | Exercice 01 : cat | 6 |
| V | Exercice 02 : tail | 7 |
| VI | Exercice 03 : hexdump | 8 |
| VII | Exercice 04 : last | 9 |

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 spécifiés 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.

Chapitre II

Préambule

Voici la recette de l'Ours à la bière :

Temps de préparation : environ 30 minutes

Temps de cuisson : 90 minutes

Ingrédients (Pour 4 personnes) :

- 1 kg de viande d'ours (épaule, idéalement)
- 5 gros oignons
- Quelques branches de thym
- 5 feuilles de laurier
- 2 betteraves crues, épluchées et coupées en quartiers
- 800 mL de bière brune
- 1 bonne grosse poignée de farine
- Huile
- Sel
- Poivre


Préparation:

- Couper la viande en cubes.
- Faire rôtir la viande dans une cocotte, avec un peu d'huile.
- Saler et poivrer.
- Réserver la viande.
- Dans la même cocotte, faire rôtir les oignons avec le thym, le laurier et les betteraves à feu moyen.
- Saupoudrer de farine, puis déglacer avec la bière.
- Rajouter la viande.
- Couvrir, puis cuire au four à 180 degrés pendant environ 1h.
- Servir chaud !

Le sujet d'aujourd'hui n'a rien à voir avec l'Ours à la bière.

Chapitre III

Exercice 00 : display_file


| | |
|---|---------------|
|  | Exercice : 00 |
| display_file | |
| Dossier de rendu : <i>ex00/</i> | |
| Fichiers à rendre : Makefile, et les fichiers de votre programme | |
| Fonctions Autorisées : close, open, read, write | |
| Remarques : n/a | |

- Écrire un programme appelé `ft_display_file` qui affiche sur la sortie standard uniquement le contenu du fichier passé en argument.
- Le répertoire de rendu aura un `Makefile` avec une règle `all`, une règle `clean`, et une règle `fclean`. Le binaire s'appellera `ft_display_file`.
- La fonction `malloc` est interdite. Vous pouvez faire l'exercice uniquement en déclarant un tableau de taille fixe.
- Tous les fichiers passés en paramètre seront valides.
- Les messages d'erreurs devront être affichés sur la sortie leur étant réservée.

```
$> ./ft_display_file
File name missing.
$> ./ft_display_file Makefile
*contenu du Makefile*
$> ./ft_display_file Makefile display_file.c
Too many arguments.
$>
```

Chapitre IV

Exercice 01 : cat


| | |
|--|---------------|
|  | Exercice : 01 |
| cat | |
| Dossier de rendu : <i>ex01/</i> | |
| Fichiers à rendre : <code>Makefile</code> , et les fichiers de votre programme | |
| Fonctions Autorisées : <code>close</code> , <code>open</code> , <code>read</code> , <code>write</code> | |
| Remarques : n/a | |

- Écrire un programme appelé `ft_cat` qui réalise le même travail que la commande `cat` du système.
- Vous n'avez pas à gérer les options.
- Le répertoire de rendu aura un `Makefile` avec une règle `all`, une règle `clean`, et une règle `fclean`.
- Vous pouvez utiliser la variable `errno` (voir le `man` de `Errno`).
- Vous pouvez faire l'exercice uniquement en déclarant un tableau de taille fixe. Ce tableau aura une taille limitée à un peu moins d'environ 30 ko. Pour que vous puissiez tester cette limitation, utilisez la commande `limit` dans votre shell.

```
$> limit stacksize 32
$> limit stacksize
stacksize 32 kbytes
$>
```

Chapitre V


Exercice 02 : tail

| | |
|--|---------------|
|  | Exercice : 02 |
| tail | |
| Dossier de rendu : <i>ex02/</i> | |
| Fichiers à rendre : Makefile , et les fichiers de votre programme | |
| Fonctions Autorisées : <code>close</code> , <code>open</code> , <code>read</code> , <code>write</code> , <code>malloc</code> , <code>free</code> | |
| Remarques : n/a | |

- Écrire un programme appelé `ft_tail` qui réalise le même travail que la commande `tail` du système mais qui prend au moins un fichier en argument.
- Vous avez à gérer uniquement l'option `-c`, cette option sera fournie dans tous les tests.
- Le répertoire de rendu aura un **Makefile** avec une règle `all`, une règle `clean`, et une règle `fclean`.
- Vous pouvez utiliser la variable `errno` (voir le `man` de `errno`).

Chapitre VI


Exercice 03 : hexdump

| | |
|--|---------------|
|  | Exercice : 03 |
| hexdump | |
| Dossier de rendu : <i>ex03/</i> | |
| Fichiers à rendre : Makefile , et les fichiers de votre programme | |
| Fonctions Autorisées : <code>close</code> , <code>open</code> , <code>read</code> , <code>write</code> , <code>malloc</code> , <code>free</code> | |
| Remarques : n/a | |

- Écrire un programme appelé `ft_hexdump` qui réalise le même travail que la commande `hexdump` du système.
- Vous n'avez à gérer que l'option `-C`.
- Le répertoire de rendu aura un **Makefile** avec une règle `all`, une règle `clean`, et une règle `fclean`.
- Vous pouvez utiliser la variable `errno` (voir le `man` de `errno`).

Chapitre VII

Exercice 04 : last

| | |
|--|---------------|
|  | Exercice : 04 |
| last | |
| Dossier de rendu : <i>ex04/</i> | |
| Fichiers à rendre : Makefile , et les fichiers de votre programme | |
| Fonctions Autorisées : <code>close</code> , <code>open</code> , <code>read</code> , <code>write</code> , <code>malloc</code> , <code>free</code> | |
| Remarques : n/a | |

- Écrire un programme appelé `ft_last` qui réalise le même travail que la commande `last` du système, sans aucune option.
- Le répertoire de rendu aura un **Makefile** avec une règle `all`, une règle `clean`, et une règle `fclean`.
- Vous avez uniquement le droit d'`include` `<utmp.h>`, `<time.h>`, `<errno.h>` et `<sys/types.h>`
- Vous ne pouvez pas utiliser les fonctions `ctime()`, `asctime()`, `stat()`, etc.