



Piscine C

C 10

Résumé: Ce document est le sujet du module C 10 de la piscine C de 42.

Version:

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 | Rendu et peer-evaluation | 9 |

Chapitre I

Consignes

- Seule cette page servira de référence : ne vous fiez pas aux bruits de couloir.
- Relisez bien le sujet avant de rendre vos exercices. A tout moment le sujet peut changer.
- 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**.
- 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.
- L'utilisation d'une fonction interdite est un cas de triche. Toute triche est sanctionnée par la note de -42.
- Vous ne devrez rendre une fonction `main()` que si nous vous demandons un programme.
- 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

vosre voisin de gauche.

- Votre manuel de référence s'appelle **Google / man / Internet /**
- Pensez à discuter sur le forum Piscine de votre Intra, ainsi que sur le slack de votre Piscine !
- 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


Pour bien commencer votre journée, voici quelques questions très simples :

- Que se passerait-il si je laissais allumé un sèche-cheveux alimenté en continu dans un cube étanche d'un mètre de côté ?
- Déverser de l'antimatière dans le réacteur de Tchernobyl quand il était en train de fondre aurait-il empêché sa fusion ?
- C'est possible de pleurer au point de se déshydrater ?
- Si tous les êtres humains disparaissaient de la surface du globe, au bout de combien de temps s'éteindrait la dernière source de lumière artificielle ?
- C'est vraiment dangereux de se baigner dans une piscine pendant un orage ?
- De quelle hauteur faudrait-il laisser tomber un steak pour qu'il soit cuit en arrivant au sol ?
- Quand la bande passante d'Internet dépassera-t-elle celle de FedEx, si elle y parvient un jour ?
- Combien de tweets différents sont possibles dans notre langue ? Et combien de temps faudrait-il à la population mondiale pour tous les lire à haute voix ?
- Quel serait le résultat si tous les candidats au code de la route répondaient au pif au questionnaire à choix multiple ? Combien répondraient juste à l'ensemble des questions ?

Questions extraites du livre 'Et si ...?' de Randall Munroe.

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 : <code>close</code> , <code>open</code> , <code>read</code> , <code>write</code> | |

- 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 et en étant suivi d'un retour à la ligne.

- Si il n'y aucun argument, votre programme doit afficher

```
File name missing.
```

- Si il y a trop d'argument, votre programme doit afficher


```
Too many arguments.
```

- Si le fichier n'est pas lisible, votre programme doit afficher

```
Cannot read file.
```

Chapitre IV


Exercice 01 : cat

| | |
|--|---------------|
|  | Exercice : 01 |
| cat | |
| Dossier de rendu : <i>ex01/</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>strerror</code> , <code>basename</code> | |

- 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 devriez aller lire les `man` de toutes les fonctions autorisées
- 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 `ulimit` dans votre shell.

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> , <code>strerror</code> , <code>basename</code> | |

- Écrire un programme appelé `ft_tail` qui réalise le même travail que la commande `tail`.
- Vous avez à gérer uniquement l'option `-c`, mais vous n'avez pas à gérer le `'+'` et le `'-'`.
- Tous les tests seront effectué avec 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`

Chapitre VI

Exercice 03 : hexdump

| | |
|--|---------------|
|  | Exercice : 03 |
| hexdump | |
| Dossier de rendu : <i>ex03/</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> , <code>malloc</code> , <code>free</code> , <code>strerror</code> , <code>basename</code> | |

- Écrire un programme appelé `ft_hexdump` qui réalise le même travail que la commande `hexdump` du système sans redirection.
- 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

Rendu et peer-evaluation

Rendez votre travail sur votre dépôt `Git` comme d'habitude. Seul le travail présent sur votre dépôt sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

Vu que votre travail ne sera pas évalué par un programme, organisez vos fichiers comme bon vous semble du moment que vous rendez les fichiers obligatoires et respectez les consignes du sujet.



`Vous ne devez rendre uniquement les fichiers demandés par le sujet de ce projet.`