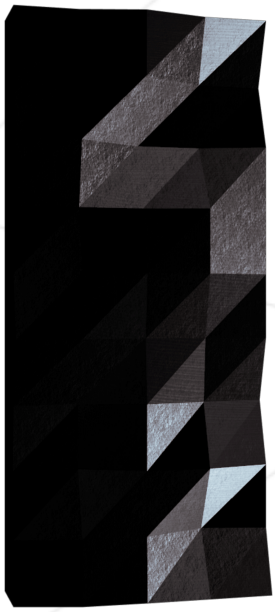

ft_ls

Aussi simple que de lister les fichier d'un répertoire

Equipe pedago pedago@staff.42.fr

Résumé: Ce projet a pour but de vous faire recoder la commande "ls".

Table des matières

I	**Préambule**	**2**
II	**Introduction**	**3**
III	**Objectifs**	**4**
IV	**Consignes générales**	**5**
V	**Partie obligatoire**	**7**
VI	**Partie bonus**	**8**
VII	**Rendu et peer-évaluation**	**9**

Chapitre I

Préambule

Voici une recette de `Choucroute à l'alsacienne` :

- Ingrédients (4 personnes)
  - 1kg de choucroute
  - 350g de lard fumé
  - 350g de palette
  - 1 càs de saindoux
  - 2 gousses d'ail
  - 1 feuille de laurier
  - 10 grains de genièvre
  - 1 oignon piqué avec 2 clous de girofle
  - 25cl de Riesling
  - 350g de pommes de terre
  - 4 saucisses de Strasbourg
- Préparation
  - Rincer la choucroute sous l'eau froide. L'égoutter, et en verser la moitié dans un faitout.
  - Incorporer le lard fumé et la palette, puis recouvrir du reste de choucroute.
  - Ajouter le saindoux, l'ail (Non pelé !), le genièvre, le laurier et l'oignon piqué.
  - Arroser de Riesling, et laisser cuire à couvert et à feu doux pendant 1 heure.
  - Ajouter les pommes de terre, et poursuivre la cuisson 50 minutes.
  - Incorporer les saucisses, puis laisser cuire encore 10 minutes.
  - Servir avec une quantité déraisonnable de bière.

`Ce projet est plus facile si vous le réalisez après avoir mangé de la Choucroute à l'alsacienne.`

Chapitre II

Introduction

La commande `ls` est une des toutes premières commandes que vous avez découvertes pendant vos premières heures de shell. Mais c'est également une de celles que vous utilisez le plus. Peut-être vous êtes vous déjà demandé comment cette commande était codée ? Quelque soit votre réponse, vous allez bientôt le découvrir grâce à ce projet.

Recoder la commande `ls` et certaines de ses options vous permettra de découvrir comment intéragir avec le système de fichiers depuis un programme en `C`. Après tout, vous savez déja comment ouvrir, lire, écrire et fermer un fichier. Mais qu'en est-il des répertoires ? Des fichiers spéciaux ? Des droits, des dates ou de la taille des fichiers ?

Ah, et pendant que j'y suis, la qualité de votre `libft` sera déterminante entre une expérience agréable et détestable sur ce projet. Par exemple, avoir déja réalisé le projet `ft_printf` et ajouté cette fonction à votre `libft` vous simplifiera la vie. Le projet `ft_ls` est tout à fait réalisable sans, de la même manière qu'on peut tout à fait manger un yaourt avec ses doigts. Mais avoir une cuillère aide quand même beaucoup...

Chapitre III

Objectifs

Le projet `ft_ls` vous ouvre la voie de l'arc `Unix` du graph de projets. Vous allez pour la première fois être confrontés à des fonctions de la `libc` vous permettant autre chose que lire ou écrire sur un file descriptor (pour caricaturer). Vous allez découvrir un sous ensemble de fonctions de l'API de votre système d'exploitation et les structures de données associées, ainsi que la gestion des allocations mémoires qui vont avec ces données.

`ft_ls` est également une excellente opportunité de réflechir à la structure de votre code avant de vous jeter à corps perdu dans l'écriture du code. La mauvaise réputation de `ft_ls` est principalement répandue par des étudiants découvrant trop tard que leur (manque de) design initial les empêche de terminer le projet sans refactoriser une immense partie de leur code. Je reconnais qu'il y a de quoi être frustré...

Pour terminer, encore et toujours, `ft_ls` sera pour vous une occasion supplémentaire de compléter votre `libft` avec de nouvelles fonctions toujours plus pratiques. Parcourir un dossier et identifier des fichiers est une tache très commune en programmation et vous serez bien entendu amenés à le faire à nouveau sur de nombreux autres projets. Améliorer votre `libft` dans ce sens sera autant de temps de gagné pour la suite.

Chapitre IV

Consignes générales

- Ce projet ne sera corrigé que par des humains. Vous êtes donc libres d'organiser et nommer vos fichiers comme vous le désirez, en respectant néanmoins les contraintes listées ici.
- L'exécutable doit s'appeller `ft_ls`.
- Vous devez rendre un Makefile. Ce Makefile devra compiler le projet, et doit contenir les règles habituelles. Il ne doit recompiler le programme qu'en cas de nécessité.
- Si vous êtes malin et que vous utilisez votre biliothèque `libft` pour votre `ft_ls`, vous devez en copier les sources et le `Makefile` associé dans un dossier nommé `libft` qui devra être à la racine de votre dépôt de rendu. Votre `Makefile` devra compiler la librairie, en appelant son `Makefile`, puis compiler votre projet.
- Votre projet doit être à la Norme.
- Vous devez gérer les erreurs de façon raisonnée. En aucun cas votre programme ne doit quitter de façon inattendue (Segmentation fault, etc...). Si vous n'êtes pas sûr, autant gérer les erreurs comme `ls`.
- Vous devez rendre, à la racine de votre dépôt de rendu, un fichier `auteur` contenant votre login suivi d'un '\n' :

```
$>cat -e auteur
xlogin$
```

- Dans le cadre de votre partie obligatoire, vous avez le droit d'utiliser les fonctions suivantes :
  - `write`
  - `opendir`
  - `readdir`
  - `closedir`
  - `stat`
  - `lstat`
  - `getpwuid`
  - `getgrgid`
  - `listxattr`

  - `getxattr`
  - `time`
  - `ctime`
  - `readlink`
  - `malloc`
  - `free`
  - `perror`
  - `strerror`
  - `exit`

- Vous avez l'autorisation d'utiliser d'autres fonctions dans le cadre de vos bonus, à condition que leur utilisation soit dûment justifiée lors de votre correction. Par exemple, utiliser `tcgetattr` est justifiable dans certains cas, utiliser `printf` par flemme ne l'est jamais. Soyez malins.
- Vous pouvez poser vos questions sur le forum.

Chapitre V

Partie obligatoire

- Vous devez recoder la commande `ls` du système.
- Son comportement doit être identique à celui de la commands `ls` originale du système, avec les bémols suivants :
  - Parmi les nombreuses options disponibles sur la ligne de commande, il vous est demandé de réaliser les suivantes : `-l`, `-R`, `-a`, `-r` et `-t`.
  - **Nous vous recommandons lourdement de prendre en compte les implications de l'option `-R` dès le début de votre code...**
  - Vous n'avez pas à gérer le formatage en plusieurs colonnes de la sortie quand l'option `-l` n'est pas passée.
  - Vous n'êtes pas obligés de gérer les `ACL` et les attributs étendus.
  - L'affichage général, selon chaque option, doit rester sensiblement identique à celui de la commande système. Une certaine tolérance est appliquée sur le padding et la mise en page, mais il ne doit manquer aucune information.

```
man ls
```

Chapitre VI

Partie bonus

Les bonus ne seront évalués que si votre partie obligatoire est EXCELLENTE. On entend par là qu'elle est entièrement réalisée, que votre gestion d'erreur est au point, même dans des cas vicieux, ou des cas de mauvaise utilisation. Dans le cas contraire, vos bonus seront intégralement IGNORÉS.

Voici quelques idées de bonus intéressants à réaliser, voire même utiles. Vous pouvez évidemment ajouter des bonus de votre invention, qui seront évalués à la discrétion de vos correcteurs.

- Gestion des ACL et des attributs étendus
- Gestion des colonnes sans l'option `-l`. (man 4 tty)
- Gestion des options `-u`, `-f`, `-g`, `-d`, ...
- Gestion d'affichage en couleur (Similaire à l'option `-G`)
- Optimisation de votre code (Quel est le temps de réponse de votre ls sur un GROS ls -lR par exemple ?)

Chapitre VII

Rendu et peer-évaluation

Rendez-votre travail sur votre dépot `GiT` comme d'habitude. Seul le travail présent sur votre dépot sera évalué en soutenance.