Minishell

Aussi beau qu'un shell

Résumé :

Ce projet consiste à créer un shell simple.

Oui, votre propre petit bash.

Vous apprendrez beaucoup sur les processus et les descripteurs de fichiers.

Version : 7.1

Sommaire

Chapitre I : Introduction

L'existence des shells est liée à l'existence même de l'informatique.

À l'époque, tous les développeurs s'accordaient à dire que communiquer avec un ordinateur en utilisant des commutateurs alignés 1/0 était sérieusement agaçant.

Il était donc logique qu'ils aient eu l'idée de créer un logiciel permettant de communiquer avec un ordinateur en utilisant des lignes de commandes interactives dans un langage relativement proche du langage humain.

Grâce à Minishell, vous pourrez voyager dans le temps et revenir aux problèmes auxquels les gens étaient confrontés avant l'existence de Windows.

Chapitre II : Instructions générales

- Votre projet doit être écrit en C.
- Il doit respecter les règles de la Norme. Si vous avez des fichiers/fonctions bonus, ils doivent être inclus dans la vérification de la norme et vous recevrez un 0 s'il y a une erreur de norme.
- Vos fonctions ne doivent pas se terminer de manière inattendue (segmentation fault, bus error, double free, etc.) à l'exception des comportements indéfinis. Si cela se produit, votre projet sera considéré comme non fonctionnel et recevra un 0 lors de l'évaluation.
- Toute mémoire allouée sur le tas doit être correctement libérée si nécessaire. Aucune fuite de mémoire ne sera tolérée.
- Si le sujet l'exige, vous devez soumettre un Makefile qui compilera vos fichiers source avec les flags -Wall, -Wextra et -Werror, en utilisant cc. Votre Makefile ne doit pas relinker.
- Votre Makefile doit contenir au moins les règles suivantes : $(NAME), all, clean, fclean et re.
- Pour soumettre des bonus à votre projet, vous devez inclure une règle bonus dans votre Makefile, qui ajoutera tous les en-têtes, bibliothèques ou fonctions qui sont interdits dans la partie principale du projet. Les bonus doivent être dans un fichier séparé _bonus.{c/h} sauf indication contraire.
- Si votre projet vous permet d'utiliser votre libft, vous devez copier ses sources et son Makefile associé dans un dossier libft. Le Makefile de votre projet doit compiler la bibliothèque en utilisant son Makefile, puis compiler le projet.
- Nous vous encourageons à créer des programmes de test pour votre projet, même si ce travail ne doit pas être soumis et ne sera pas noté. Cela vous permettra de tester facilement votre travail et celui de vos pairs.
- Soumettez votre travail dans le dépôt git qui vous est attribué. Seul le travail présent dans ce

dépôt sera noté. Si Deepthought est chargé d'évaluer votre travail, cela se fera après l'évaluation par vos pairs.

Chapitre III : Partie obligatoire

- Nom du programme : minishell
- Fichiers à rendre : Makefile, *.h, *.c
- Arguments : aucun
- Fonctions externes : readline, add_history, printf, malloc, free, write, fork, execve, waitpid, signal, kill, exit, getcwd, chdir, stat, dup2, pipe, opendir, readdir, closedir, etc.
- Libft autorisé : oui
- Description : Écrire un shell capable d'afficher une invite, d'exécuter des commandes, de gérer des redirections (<, >, >>), des pipes (|), les variables d'environnement ($VAR), et les signaux (Ctrl+C, Ctrl+D, Ctrl+\). Votre shell doit implémenter les commandes internes : echo, cd, pwd, export, unset, env, et exit.

Chapitre IV : Partie bonus

- Implémenter && et || avec des parenthèses pour la priorité.
- Gérer les jokers (*) pour le répertoire de travail courant.

Important : La partie bonus ne sera évaluée que si la partie obligatoire est parfaitement réalisée.

Chapitre V : Soumission et évaluation par les pairs

Soumettez votre travail dans votre dépôt Git comme d'habitude. Seul le travail dans le dépôt sera évalué lors de la défense. N'hésitez pas à vérifier les noms de vos fichiers pour vous assurer qu'ils

sont corrects.