PROJET : MiniSHell « my_sh » – à réaliser en binôme – Durée approximative ~ 8h00min.

A l'aide des supports de cours et des mémentos et des exercices achevés en TP et du manuel Linux réalisez :
(Tout code ou implémentation compilant ou non sera étudié, toute fois le barème sera adapté)
Le présent sujet de projet comporte 3 pages

*I – **Synopsis***

L'objectif de ce projet et de réaliser *-dans une proportion simplifiée-* l'implémentation d'un interpréteur de commande similaire à **bash**. Au lancement ce dernier doit pouvoir afficher un prompt attendant la saisie d'une commande ou d'un sous-ensemble de commandes.

Exemple :

***Prompt**> ls –a **;** who* ***Prompt**> ls –a **|** grep toto*
***Prompt**> date*

*II – **Démarche***

Les commandes sont lancées par des processus fils laissant au père le rôle d'interpréteur. Ce dernier doit attendre la fin du/des processus fils pour afficher le résultat d'exécution du/des commande(s) soumises. Certaines commandes et variables devront être internes (fonctionnalités *built-in),* c'est-à-dire, directement prises en compte par le code.

Ainsi on distinguera trois phases distinctes :

1. L'évaluation de l'expression soumise à l'interpréteur.
2. L'exécution ordonnancée du sous-ensemble de commandes.
3. La soumission du résultat d'exécution.

*III – **Résultats attendus***

Le livrable attendu pour ce projet se résume en un code source **compilable** et **exécutable** répondant d'une part aux fonctions métiers suivantes :

- **FM01** – Le binaire est capable d'exécuter une commande simple (*ie :* ls –l ; ps ; who)
- **FM02** – Le binaire est capable d'exécuter un sous-ensemble de plusieurs commandes de sorte à prendre en compte :
  - Les opérateurs de contrôle *:* **&&** *et* **||**
  - Les redirections de flux simples : **|**, **>, <, >>, <<**
  - L'exécution en arrière-plan : **&**
- **FM03** – L'exécution des commandes internes (fonctionnalités built-in) suivantes :
  - **cd** - Permettant de se déplacer au sein d'une arborescence de fichier.
  - **pwd** – Affichant la valeur de la variable contenant le chemin du répertoire courant.
  - **exit** – Permettant de quitter l'interpréteur.
  - **echo** – Permettant d'afficher du texte sur la sortie standard.
- **FM04** - La persistance des commandes saisie dans un fichier (historique)

D'autres fonctionnalités optionnelles peuvent êtres implémentés :

- **FO01** – La réalisation d'un mode batch (ie : ./my_shell –c « ls –al | grep toto »)
- **FO02** – La création de variables d'environnement
- **FO03** – La prise en charge d'alias

Concernant les exigences techniques attendues, vous devez respecter les contraintes suivantes :

- **CT01** – La compilation du projet doit se faire via un Makefile.
- **CT02** – La définitions des structures doit se faire dans un fichier typedef.h.
- **CT03** – La définition des méthodes protoype (*.h*) & implémentation (*.c*) doit se faire de manière séparée autant que faire se peut.
- **CT04** – Le code produit doit être documenté.
- **CT05** – La gestion des erreurs doit se faire via « les mécanismes proposés par errno ».

D'autres contraintes techniques peuvent être prises en compte :

- **CTO01** – La documentation du code générée via l'utilitaire *doxygen.*
- **CTO02** – Le code est soumis à un contrôle de couverture via l'utilitaire *gcov.*
- **CTO03** – Une page de manuel Linux est rédigée pour détailler l'exécution du shell.

*IV – **Evaluation***

Ce projet est à réaliser en binôme et donnera lieu à une présentation d'environs 15min répartie en deux phases : *présentation&démonstration* suivi d'un partie consacrée aux *questions*. *(Un support de présentation pourra être utilisé mais ne devra pas contenir plus de 5 diapos).*

La réalisation de ce shell simplifié vise à mettre en œuvre l'ensemble des connaissances abordées au cours de ce module. La restitution du plus grand nombre de notions à travers le code produit vous permet de valider vos compétences.

Ainsi la réalisation de l'ensemble des fonctionnalités métiers **FM 1** à **4** ainsi que le respect des contraintes techniques vous assure une note supérieure à la moyenne signifiant l'acquis des connaissances. Toutefois la réalisation des fonctionnalités métiers et/ou optionnelles vous permettent d'augmenter votre note le cas échéant.

*IV – **Rappels***

Approche incrémentale du développement

Pour parvenir au résultat attendu, veuillez toujours appliquer une approche incrémentale en termes d'ajout de code/fonctionnalité, procédez par étape afin de ne pas avoir un code C trop complexe qui serait *in-fine* difficile à débugger.

Par exemple: une approche incrémentale pour ce type d'exercice « ls-like » serait :

1. La récupération des paramètres
2. Tester fichier/répertoire
3. Parcourir les éléments du répertoire en affichant leur nom
4. Alimenter chaque fichier avec une information supplémentaire : permission / taille / propriétaire…

Documentation

Pour obtenir des informations ou de la documentation ayez le réflexe d'utiliser les pages du manuel.

Par exemple:

- man 3 stat / man 2 open / man 2 readdir / man errno

Limitation d'usage

Vous pouvez vous servir de n'importe quel librairie tierce pour réaliser l'implémentation de vos algorithmes et /ou structures dans la mesure où elle n'implémente pas directement une fonctionnalité métier.

Gestion des erreurs

Afin d'avoir une gestion des erreurs la plus précise possible ayez le réflexe d'utiliser les codes retours ***ERRNO*** spécifiés dans les pages de manuel.

Par exemple:

- **EEXIST** `File exists (POSIX.1)`
- **EFAULT** `Bad address (POSIX.1)`
- **EISDIR** `Is a directory (POSIX.1)`
- **ENOTDIR** `Not a directory (POSIX.1)`
- **ELOOP** `Too many levels of symbolic links (POSIX.1)`