Interpréteur de commandes UNIX simplifié

TP : Provisoire

Contenu

Objectif

L'objectif de ce projet est d'implémenter un interpréteur de commandes Unix simplifié en C, similaire à bash.

La première partie du travail est composé de deux points :

- Lire et d'écrire des variables d'environnement.
- Naviguer dans le système de fichier et créer des alias.

Variables d'environnement

- ***Lecture du fichier profile :***

```
void read_profile_file () ;
```

Dans un premier temps, la fonction récupère le répertoire de travail de l'utilisateur, nous l'avons cherché avec la fonction :

```
struct passwd *getpwuid(uid_t uid);
```

Cette fonction demande en paramètre le « User ID », il est récupérer avec la fonction

```
uid_t getuid(void);
```

Dans la structure passwd, nous récupérons l'information du répertoire de travail (pw_dir) :

```
struct passwd {
      char   *pw_name;       /* username */
      char   *pw_passwd;     /* user password */
      uid_t   pw_uid;        /* user ID */
      gid_t   pw_gid;        /* group ID */
      char   *pw_gecos;      /* user information */
      char   *pw_dir;        /* home directory */
      char   *pw_shell;      /* shell program */
};
```

Maintenant nous pouvons ouvrir (ou si inexistant créer) le fichier profile avec open(), pour ainsi récupérer le fd (file descriptor), utile à la lecture du fichier avec read().

```
int fd_profile = open(strcat(p->pw_dir,"/profile"),
O_RDWR | O_CREAT, S_IRWXU);

read(fd_profile, buffer_str, BUFFER_STR);
```

Le texte du fichier est ensuite découper avec strtok() et comparer avec strncmp() pour vérifier si PATH et HOME sont existant. Si ce n'est pas le cas, ils sont écrits dans le fichier.

A la fin de la fonction ils sont ajoutés comme variable d'environnement.

- ***Ajout ou modification d'une variable d'environnement :***

```
int add_env(const char* name_value);
```

La syntaxe d'entrée de notre fonction est : « NAME=VALUE »

Il est découper en deux morceau avec strtok(), comme séparateur le premier caractère '=' de la chaine. Ensuite donné en paramètre à setenv() .

```
int setenv(const char *name, const char *value, int
overwrite);
```

- ***Lecture d'une variable d'environnement :***

```
char* read_env(const char* name);
```

La syntaxe d'entrée de notre fonction est : « $NAME »

read_env enlève le caractère '$' du nom et le donne en paramètre à getenv()

```
char* getenv(const char* name);
```

Qui lui retourne null si la variable n'existe pas.

Naviguer dans le système de fichier et créer des alias

- ***Changer le répertoire de travail :***

```
int cd(const char* directory) ;
```

La syntaxe d'entrée par exemple : « /tmp »

Il envoie ce paramètre à chdir()

```
int chdir(const char* path);
```

Qui lui retourne variable d'erreur, errno

- ***Afficher le répertoire de travail courant :***

```
char* pwd() ;
```

Avec la fonction getcwd() il est possible de récupérer le répertoire de travail courant.

```
char* getcwd(char* buf, size_t size);
```

Pour configurer la taille du buffer nous avons utilisé pathconf() :

```
Long size = pathconf(".", _PC_PATH_MAX);
```

- ***Assigner un alias :***

Chaque alias est ajouté à un tableau de structure :

```
/**
 * Structure alias
 */
typedef struct alias alias_t;
struct alias
{
      char* name;
      char* value;
};
```

Une commande d'ajout d'un alias est d'abord recherchée, s'il est trouvé uniquement la valeur dans la structure est modifiée. Sinon un nouvel alias est ajouté avec le nom et valeur.

```
void add_alias(alias_t* alias, const char* name_value);
```

Il est aussi possible de voir tous les alias créer avec l'option : alias -p

RENDU PROVISOIR ------- Prochainement complet !