INF203 - Exercices semaine 11

Interpréteur de commandes

Pour les exercices qui suivent, vous disposez de l'ensemble de fonctions suivant permettant de manipuler des variables stockées dans un ensemble :

```
/* initialisation d'un ensemble de variables a l'ensemble vide */
void init_variables(variables *ens);

/* ajout d'une variable dans un ensemble de variables si elle n'y est pas deja,
 * ecrase sa valeur sinon. Dans tous les cas renvoie l'indice de la variable dans
 * l'ensemble */
int ajouter_variable(variables *ens, char *nom, char *valeur);

/* cardinal d'un ensemble de variables */
int nombre_variables(variables *ens);

/* recherche de l'indice d'une variable dans un ensemble, -1 si la variable n'est
 * pas presente dans l'ensemble */
int trouver_variable(variables *ens, char *nom);

/* variable d'indice i (NULL si cette variable n'existe pas dans l'ensemble) */
char *nom_variable(variables *ens, int i);

/* valeur de la variables a l'indice i ("" si inexistant) */
char *valeur_variable(variables *ens, int i);

/* met a jour la valeur de la variable d'indice i (sans effet si indice invalide) */
void modifier_valeur_variable(variables *ens, int i, char *valeur);
```

Exercice 1 :

Écrivez une fonction de prototype :

```
void afficher_ensemble_variables(variables *ens);
```

Permettant d'afficher le contenu de l'ensemble de variables `ens` avec un couple (nom, valeur) par variable et par ligne sous la forme nom=valeur.

Exercice 2 :

Écrivez une fonction de prototype :

```
int trouver_et_appliquer_affectation_variable(variables *ens, char *ligne);
```

Permettant de déterminer si la chaîne de caractères `ligne` est de la forme `identificateur=valeur` :

- si c'est le cas, cette fonction devra affecter, dans l'ensemble `ens`, la valeur de la variable `identificateur` à la valeur donnée. Si la variable `identificateur` existe déjà, cette fonction devra écraser la valeur existante. Ensuite, la fonction devra retourner 1 ;
- si ce n'est pas le cas, de ne rien faire et de retourner 0.

Exercice 3 :

Écrivez une fonction de prototype :

```
void appliquer_expansion_variables(variables *ens, char *ligne_originale,
                                   char *ligne_expansee);
```

Permettant de copier dans `ligne_expansee` le contenu de la chaîne de caractères `ligne_originale` en remplaçant toutes les occurences d'une sous-chaîne de la forme `$identificateur` par la valeur de la variable de nom `identificateur` dans l'ensemble `ens`. Les caractères formant un nom de variable sont les caractères `c` tels que `isalnum(c)` est vrai.

Exercice 4 :

Écrivez une fonction de prototype :

```
void decouper_ligne(char *ligne, char *ligne_decoupee[]);
```

Permettant de découper la chaîne de caractères `ligne` et de stocker des pointeurs vers les mots de `ligne` dans `ligne_decoupee`. Le découpage se fera par modification du contenu de `ligne` en stockant un caractère de fin de chaîne à la place de chaque séparateur suivant immédiatement un mot. `ligne_decoupee` devra contenir un pointeur vers la première lettre de chaque mot contenu dans `ligne` et se terminer par `NULL`.

Exercice 5 :

Reprenez la question 3 en tenant compte des mécanismes du shell permettant de protéger contre l'expansion (' et \).