Université de Rouen
I.U.P. 2
Module : Système I

T.P. 11 : Files de messages et verrous

Exercice 1 Soit $\Sigma = \{a_1, a_2, \ldots, a_k\}$ un alphabet à k lettres. Soit L_m un ensemble de mots de longeur m sur l'alphabet Σ.
Exemple : $\Sigma = \{a, b, c\}$ et $L_4 = \{\ aabc,\ abbc,\ ccaa\ , cbca\ , aaac\ \}$. Dans cet exemple nous avons $a_1 = a$, $a_2 = b$ et $a_3 = c$. Ainsi on peut faire correspondre à chaque lettre de l'alphabet un indice.
Le problème est d'implémenter un tri lexicographique (tri du dictionnaire) en utilisant les files de messages.

Algorithme

Debut
On construit $k + 1$ listes.
On lit les NB_MOT mots dans la liste $k + 1$.
Pour i:=m **a** 1 **faire**
Debut
Pour chacun des mots de la liste k+1 **faire**
Debut
On place le mot considere dans
la liste de numero l'indice de la i^{eme} lettre
Fin
On concatene les k premieres listes dans la liste $k + 1$.
Fin
Ecrire la liste $k + 1$.
Fin

Remarque : Toute lecture dans une liste est destructive.
On pourra n'utiliser qu'une seule liste.

Exercice 2 On désire créer et gérer une liste du personnel d'une entreprise. Cette liste sera stockée dans un fichier Unix sous forme de liste de structure. La structure comprendra les informations suivantes:

- Numéro d'identification,
- nom
- prénom,
- date de naissance,
- nom du service,
- qualité,
- salaire.

Un client pourra mettre à jour, insérer, supprimer ou rechercher des informations dans cette liste.

Créer une fonction par opération demandée.

```
int maj(identifiant, nom-du-champ, valeur);
int insere(nom, prenom, date-de naissance, ...);
int supprime(identifiant);
int recherche(nom-du-champ, valeur);
int liste();
```

Les fonctions `insere` et `recherche` retourne un numéro d'identifiant. Les autres renvoie 0 si tout c'est bien passée, -1 sinon.

D'autre part, il sera possible de lancer plusieurs clients de façon concurrente. Vous devrez tenir compte de ce paramètre et gérer les problèmes d'accès concurrents à l'aide des verrous POSIX disponible sous Unix.

Vous utiliserez pour ce faire la fonction :

```
int fcntl(desc, commande, verrou);
```

où `verrou` est un pointeur sur une structure `struct flock`.

```
struct flock
  {
    short int l_type;   /* F_RDLCK, F_WRLCK ou F_UNLCK */
    short int l_whence; /* SEEK_SET, SEEK_CUR ou SEEK_END */
    off_t l_start;      /* position relative du debut p.r.a l_whence */
    off_t l_len;        /* longueur: 0 <==> jusqu'a fin de fichier */
    pid_t l_pid;        /* pid du processus auquel appartient le verrou */
  };
```

`commande` est parmi

F_SETLK Demande de pose non bloquante de verrou.

F_SETLKW Demande de pose bloquante de verrou. Si il existe déja un verrou bloquant, le processus est endormi jusqu'à ce qu'il n'y est plus de verrou incompatible ou que l'appel soit interrompu.

F_GETLK Test d'existence d'un verrou incompatible avec le verrou donné. S'il n'existe pas, le champ `l_type` de `verrou` vaut `F_UNLCK` sinon, `verrou` est rempli avec les informations sur ce verrou incompatible.