Informatique embarqué

TP n°2 - MION Giovanni

Temps estimé: 10 heures

Temps effectif: 11 heures

I)Compilation du code

Compilation avec liaison des dynamique librairies:
make

avant exécution faire:
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:.

Compilation avec liaison statique des librairies:
make static

Suppression des fichiers temporaires:
make clean

Suppression de l'ensemble des fichiers générés par make:
make mrproper

NB: le makefile ne fonctionne qu'avec le compilateur GCC.

II)Choix de l'implémentation

a) Les listes

Les listes sont implémentées comme spécifié sur l'énoncé du TP. Pour crée un éléments et l'ajouter dans une liste il faut ajouter à la structure de cet élément un membre de type liste_head.

Une liste est un élément de type liste_head initialisé avec INIT_LIST_HEAD prenant en paramètre sont adresse.

Pour ajouter un élément à la liste, il suffit d'utiliser la méthode list_add avec comme argument un pointeur sur le membre liste_head de cet éléments et un pointeur sur la tête de la liste.

De même, pour supprimer un élément d'une liste on peut utilisé la méthode list_del en lui passant un pointeur sur le membre list_head de l'élément.

Pour parcourir les élément d'une liste, un macro permet de définir une boucle de la forme list_for_each_entry(cur, head, member) ou cur est un pointeur sur les éléments de la liste, head est un pointeur sur la tête de la liste et membre est le nom du membre list_head dans la structure de l'élément.

b) Les tables de hachage

Les tables de hachage sont implémentées comme un tableau de liste. Il s'agit d'un tableau de 50 listes indépendantes. Les éléments à inséré dans une table de hachage doivent comporter un membre hash_table_entry où il est nécessaire de fournir un ID entier.

Une fois ce membre instancié et affecté d'un ID il peut être inséré dans une table de hachage définit par la structure hash_table avec la fonction hash_table_add en lui passant l'adresse du membre hash_table_entry de la structure ainsi que l'adresse de la table table où il doit être enregisté.

Un ajout invoque la fonction de hachage hash_entry qui à partir de l'ID déterminera l'indice de la liste ou l'élément sera ajouté.

On peut accéder a un élément de la table en utilisant la macro hash_table_get_by_id prenant en paramètre :

- un pointeur qui référencera la structure si elle est trouvée ou vaudra NULL sinon
- un pointeur vers la table de hachage
- un entier représentant l'ID de l'élément désiré
- le nom du membre du type hash_table_entry dans la structure de l'élément

La suppression d'un élément de la table se fait pas l'appel à hash_table_del sur un pointeur du membre de type hash_table_entry de l'élément à supprimer.

Il est a noter que les fonctions de hachage a partir d'un hash_table_entry ou d'un entier représentant un ID sont disponible (voir libhashtable.h).

III)Problèmes rencontrés

Voici une liste des problèmes rencontrés:

- Définition et utilisation des Macros.
- Interdépendance des librairies dans les headers

IV)Commentaire d'un autre étudiant

Après relecture du code par Kenny Pancarte voici les commentaires que j'ai conservés et qui on donner lieu à des corrections :

- Rester homogène par rapport au reste du code (utilisation de - > pour les pointeur)
- Expliquer choix de cette formule pour le calcul d'indice pour la fonction hash

En revanche le commentaire suivant n'a pas été conservé :

- Inutile de mettre node->suivant et node->precedent à NULL. On ne s'occupe que du chaînage

Car une suppression d'un élément dans la liste implique qu'il ne soit plus pointé par les autres éléments et qu'il ne pointe plus vers aucun élément de la liste pour être sur qu'il est isolé.