

TD 6 : listes chaînées

Programmation en C (LC4)

Semaine du 5 mars 2007

1 Listes simplement chaînées

On se donne une structure de liste chaînée d'entiers définie ainsi :

```
typedef struct sliste {
    unsigned int n;
    struct sliste *suivante;
} sliste_t;
```

Exercice 1 Écrire une fonction `sliste_t *liste_vide (void)` qui renvoie la valeur de pointeur que vous aurez choisie pour représenter la liste vide. Écrire la fonction `int est_vide(sliste_t *liste)` qui teste si une liste est vide.

Exercice 2 Écrire deux fonctions d'affichage des éléments d'une liste. L'une itérative, l'autre récursive.

Exercice 3 Écrire une fonction `sliste_t *ajoute(unsigned int n, sliste_t *liste)` qui renvoie une liste ayant comme premier élément l'entier `n` et dont la suite est constituée de `liste` ainsi que la fonction `sliste_t *enleve_tete(sliste_t *liste)` qui prend une liste non vide, lui enlève sa tête et renvoie la suite de la liste.

Exercice 4 Écrire une fonction `void libere_liste(sliste_t *liste)` qui libère toute la mémoire occupée par une liste.

Exercice 5 Écrire une fonction `sliste_t *enleve_element(unsigned int n, sliste_t *liste)` qui enlève la première occurrence de `n` dans la liste donnée en argument.

2 Stratégie d'allocation mémoire

On se donne une structure de données `memoire_t` pour modéliser la mémoire d'un système d'exploitation et les procédures d'allocation et de désallocation dynamiques de la mémoire.

`memoire` est un tableau d'octets correspondant à l'ensemble des octets de la mémoire. Certaines zones de ce tableau peuvent être allouées. On stocke alors dans une liste d'entiers `zones_allouees`, une description de cette zone mémoire. Lorsqu'on libère une zone mémoire, on rajoute sa description dans la liste d'entiers `zones_libres`. Au début du processus, `zones_allouees` et `zones_libres` sont vides. `taille_occupation` est l'indice du tableau d'octets à partir duquel la mémoire est vierge, c'est à dire sans zone allouée ni zone libérée.

Pour décrire une zone mémoire, on utilise un entier 32 bits dont les 16 premiers bits correspondent à son indice dans `memoire` et les 16 derniers à sa taille. Ce type de donnée est nommé `zone_t`. Voici la définition en C de ces types :

```

#define TAILLE_MEMOIRE 1024
typedef unsigned int zone_t;

typedef struct memoire {
    char memoire [TAILLE_MEMOIRE];
    int taille_occupation;
    sliste_t *zones_libres;
    sliste_t *zones_allouees;
} memoire_t;

```

Exercice 6 Écrire une fonction `zone_t zone(unsigned int adr, unsigned int t)` encodant l'indice `adr` et la taille `t` dans un entier. Écrire les deux fonctions d'extraction de la taille et de l'indice encodés dans un entier :

```

unsigned int adr_zone(zone_t zone)
unsigned int taille_zone(zone_t zone)

```

Exercice 7 Écrire une fonction

```
void ajoute_zone_libre(unsigned int adr, unsigned int t, memoire_t *mem)
```

qui ajoute dans la liste des zones libres une zone d'adresse `adr` et de taille `t`, en utilisant les fonctions définies dans la première partie. De même, écrivez une fonction

```
void ajoute_zone_allouee(unsigned int adr, unsigned int t, memoire_t *mem)
```

Exercice 8 Écrire une fonction

```
unsigned int utilise_zone_libre(unsigned int t, memoire_t *mem)
```

qui parcourt la liste des zones libres à la recherche d'une zone dont la taille est supérieure à `t`. Si elle n'en trouve pas, la fonction renvoie `TAILLE_MEMOIRE`, sinon elle enlève `t` de la taille de la zone et renvoie l'indice dans le tableau de la mémoire libre utilisable.

Exercice 9 Écrire une fonction `char *alloue_zone(unsigned int t, memoire_t *mem)` qui utilise la fonction précédente pour chercher une zone libre utilisable. S'il n'existe pas de telle zone, on alloue une zone à la fin de la mémoire utilisée pour l'instant. Si la mémoire n'est pas suffisante, le programme est arrêté. Cette fonction renvoie un `char *`, qui pointe vers le début de la zone allouée.

Exercice 10 Écrire une fonction `void libere_zone(char *ptr, memoire_t *mem)` qui cherche si `ptr` correspond bien à un pointeur sur une zone allouée. Si c'est le cas, la fonction l'enlève des zones allouées pour la placer dans la liste des zones libres.