TP2 Gestion de mémoire - Système d'exploitation

Sereysethy Touch

09 octobre 2018

Nota :
— Que les exercices marqués * sont obligatoires, vous devez rendre vos travaux avant le **19/10/2018**.
— Pour chaque exercice, il faut toujours un fichier Makefile qui permet de compiler vos programmes et éventuellement un fichier `README.md` indiquant comment exécuter votre programme.
— S'il s'agit de documenter un programme, il faut simplement un fichier `README.md`.

Exercice 1*

Étant donnée le listing du programme suivant :

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>

#define SIZE 5

int getint(int *);

int main() {
  int n, array[SIZE];

  for (n = 0; n < SIZE && getint(&array[n]) != EOF; n++)
        ;
  return 0;
}

int getint(int *pn)
{
  int c, sign;
  while (isspace(c = getchar()))
            ;
```

```
    if (!isdigit(c) && c != EOF && c != '+' && c != '-') {
      ungetc(c, stdin);
      return 0;
    }

    sign = (c == '-') ? -1 : 1;
    if (c == '+' || c == '-')
      c = getchar();

    for (*pn = 0; isdigit(c); c = getchar())
      *pn = 10 * *pn + (c - '0');
      *pn *= sign;

    if (c != EOF)
      ungetc(c, stdin);

    return c;
}
```

Répondez aux questions suivantes :

1. Que fait ce programme ?
2. En utilisant le gdb, mettez en évidence le passage par référence de chaque élément du tableau.
3. Essayez de comprendre les rapports de la commande *strace*

Votre rapport doit être rédigé dans un fichier `README.md` en markdown.

Exercice 2* - liste chaînée

On veut créer un programme de dictionnaire qui permet de stocker des mots et ainsi que leurs définitions. Il est à noter que le programme devra utiliser une liste chaînée afin qu'on peut stocker invariablement plusieurs mots. De ce fait, vous devez implémenter au moins ces fonctions suivantes :

— déclarer une structure de données d'un élément de dictionnaires.
— écrire une fonction permettant de tester si un dictionnaire est vide. La fonctionne retourne un nombre positif si le dictionnaire est vide sinon zéro dans le cas contraire.
— écrire une fonction permettant d'ajouter un mot à la tête de liste.
— écrire une fonction permettant d'ajouter un mot à la queue d'une liste.
— écrire une fonction permettant d'ajouter un mot en donnant l'indice d'insertion.
— écrire une fonction permettant de supprimer un mot à la tête de liste.
— écrire une fonction permettant de supprimer un mot à la queue de liste.
— écrire une fonction permettant de supprimer un mot en donnant son indice.

— écrire une fonction permettant de supprimer un mot en donnant le mot à supprimer.
— écrire une fonction permettant de rechercher un mot.
— écrire une fonction pour copier un dictionnaire. La fonction retourne un nouveau dictionnaire.
— écrire une fonction affichant tous les mots entrés par page de 10.
— écrire une fonction affichant un menu pour faire appels à des fonctions
— écrire une fonction permettant de trier le dictionnaire par order croissant et décroissant. en haut.
— écrire un makefile qui permet de compiler ce programme.

Lorsque le programme se termine, vous devez prendre soin de libérer toutes les zones de mémoires utilisés, si votre programme n'effectue pas proprement ce nettoyage, cela entraîne **un point de moins** sur la note attribuée à cet exercice.

Exercice 3* - liste doublement chaînée

En reprenant l'énoncé de l'exercice 2, refaire toutes les fonctions en utilisant une liste doublement chaînée. Une list doublement chaînée est une liste dont chaque élément maintient deux pointeurs qui pointent respectivement sur un élément précédent et un élément suivant.