CS228 - TP2 Listes chaînées.

Grenoble INP – Esisar – année 2023-24

Version du 16 juin 2024

Objectif pédagogique

- Pratiquer l'algorithmique et la programmation C
- Pratiquer/réviser les fiches de syntaxe C "niveau 1 et 2".
- Pratiquer l'usage de bibliothèque.

Ni le code ni les réponses ne sont évaluées, l'objectif est de **pratiquer**.

Liens vers des documents utiles:

- Fiches de syntaxe du département info : https://chamilo.grenoble-inp.fr/courses/DEPINFOESISAR/index.php
- La compilation séparée http://laure.gonnord.org/pro/teaching/ProgAvancee1011_IMA/coursMakeCompilSeparee4p.pdf

Consignes:

- Utiliser l'image "compilation" de préférence dans les salles de TP "réseaux", et favoriser l'usage du compilateur `clang` (`clang-14` sur certaines machines).
- Machines perso : installer le paquet `glib-2.0`.
- Récupérer l'archive fournie sur Chamilo, désarchiver **dans un répertoire adéquat** (et à la ligne de commande).

Partie 1 : usage de qsort/bsearch (Tableaux) (répertoire `tabs`)

Le fichier fourni compile avec des *warnings*:

- Inclure les bons fichiers d'entête.

Il est fourni une fonction de comparaison (vue en cours), adéquate pour l'utilisation de la fonction de bibliothèque `qsort`.

Directement dans le main:

- déclarer, allouer, et initialiser un tableau de 100 entiers de manière aléatoire (voir la fonction fournie `init_random_array`.
- le trier en utilisant `qsort`
- utiliser `bsearch` pour chercher un élément précis dans le tableau trié: l'appel sera de la forme (voir le manuel):

```
int * res = bsearch(xx, yy, tt, sizeof(int), compare_ints);
```

- Tester, évidemment.

Le manuel de `qsort` fournit un exemple de tri de tableaux de chaînes (de caractères):

- déclarer et initialiser un tableau **statique** de 3 chaînes de caractères.

- regarder le manuel de `qsort` et celui de `strcmp`, comprendre et copier le code de `cmpstringp`.
- trier le tableau de chaînes en utilisant `qsort`.
- vérifier en affichant 1 élément bien choisi avant et après le tri.

Partie 2 : listes chaînées "à la main" (répertoire `clists1`)

- Récupérer et observer la déclaration des listes sur la fiche de syntaxe "C niveau 2".
- Récupérer votre préparation des fonctions de listes, implémenter et tester les fonctions suivantes (voir le fichier `listes.h`):
  - affichage
  - ajout en tête
  - taille
  - test d'appartenance
  - destruction

En les testant, bien sûr.

- Critiquer le manque de commentaires du `.h` :-)

Partie 3 : utilisation de la bibliothèque `glib` (répertoire `clists2`)

Nous allons utiliser une bibliothèque dont la documentation est disponible à cette adresse: https://docs.gtk.org/glib/

- quelle est la licence de cette implémentation?
- quel est normalement l'usage de cette bibliothèque?
- chercher la (sous) documentation pour les listes simplement chaînées.

On fournit également un Makefile qui devrait suffire pour compiler. En cas de problème de chemins, la commande: `pkg-config --libs glib-2.0 -cflags` peut être utile pour trouver les chemins pour pouvoir compiler correctement.

- Regarder le Makefile. A quoi sert l'option `-I` ? l'option `lglib-2.0`

Réaliser via des appels à la bibliothèque (dans l'ordre que vous voulez):

- la construction d'une liste d'entiers 23, 1515, 42 (append)
- l'impression de cette liste
- le calcul de la taille
- le test d'appartenance
- un tri par ordre croissant sur les valeurs entières (il faudra écrire une fonction compare).

Ne pas oublier de libérer la mémoire. Vérifier avec l'outil `valgrind`.

Optionnel: en Java !

Réaliser le même exercice en Java en utilisant les `LinkedLists` et `Collections` (pour trier).