TD noté: File à priorité

5 Novembre 2015

Le travail est individuel. Il faudra rendre un fichier portant le nom de l'étudiant suivi de l'extension `.c` *(donc* `toto.c` *pour l'étudiant Toto), et le déposer sur e-campus2 dans la boîte de dépots. Le travail rendu doit compiler sans erreurs avec la commande* `gcc toto.c` *pour être évalué. Il sera tenu compte de la lisibilité de celui-ci.*

Objectif

On souhaite simuler le comportement d'un programme de gestion de requêtes adressées à une imprimante. Ce programme sert d'interface entre l'imprimante et les utilisateurs : il peut recevoir à tout moment des requêtes d'impression adressées par des utilisateurs qu'il est alors chargé de mémoriser, et des ordres de traitement adressés par l'imprimante auxquels il doit répondre en transmettant une des requêtes d'impression correspondante en attente (s'il y en a).

Concrètement, ces requêtes seront mémorisées dans une structure de données de type file d'attente :

— lors de la réception d'une requête d'impression celle-ci doit être insérée dans la file ;

— lors de la réception d'un ordre de traitement une requête doit être extraite de la file.

On s'intéressera dans cette version à une politique de gestion de la file basée sur des priorités, matérialisées par un entier (plus l'entier est grand, plus la priorité qui lui correspond est élevée) :

— lors du dépôt de la requête, l'utilisateur précise la priorité associée ;

— lors d'une demande de traitement, *la plus ancienne requête* parmi celles de plus grande priorité présentes dans la file sera extraite.

Une requête est implémentée par la structure suivante

```
struct requete{
  int numero;    // numéro de la requête
  int priorite;  // priorité de la requête
};
typedef struct requete Requete;
```

Le programme de simulation (fonctions `simulationFile1` et `simulationFile2`) qui est fourni génère aléatoirement `Nmax` requêtes d'impression. A chaque étape de la simulation, soit une requête est générée, soit un ordre de traitement est envoyé. Ce choix est aléatoire avec une probabilité uniforme.

L'objectif du TP est de programmer les fonctions de manipulation de la file à priorité (FAP) en utilisant 2 structures différentes :

— la structure `Fap1` utilise une seule liste chaînée ;

— la structure `Fap2` est un tableau dynamique de listes chaînées, une par niveau de priorité.

Les requêtes seront stockées dans le zone de mémoire dynamique.

Travail à effectuer

Un fichier `etudiant.c` est à récupérer sur e-campus et à compléter. Les deux structures sont à définir, et pour chaque structure il faut écrire les fonctions suivantes (les fonctions qui terminent par 1 correspondent à la manipulation de la structure `Fap1`, et les fonctions qui terminent par 2 correspondent à la manipulation de la structure `Fap2`) :

— initialisation de la structure ;

— affichage du contenu de la structure ;

— ajout d'une requête à la structure ;
— extraction d'un élément de la structure (pour rappel, l'élément le plus ancien parmi les requêtes de plus haute priorité) ;
— libération de la mémoire à la fin du programme.

Il est conseillé de commencer par écrire toutes les fonctions pour la structure 1 (en les testant progressivement) avant de passer à la structure 2. Les fonctions écrites pour la structure 1 pourront être utilisées pour la structure 2.

Vous instrumenterez le code afin de mesurer le coût moyen d'ajout et d'extraction d'une requête pour chaque structure. Vous indiquerez ces valeurs dans un commentaire.