Licence 2 MPCIE - Informatique
Module : Systèmes informatiques

Projet
Systèmes informatiques

Modalités et Présentation du projet

0.1 Modalités

Le projet est à réaliser **SEUL**. Un dépot sera disponible sur Moodle vous permettant de déposer une archive au format .zip contenant l'ensemble de vos sources au format prénom_NOM.zip. La date limite pour rendre le projet est le **30 Avril 2020** à 23h59. Il est toutefois possible de discuter entre vous pour régler un problème mais toute copie d'un code d'un autre étudiant sera sanctionnée. Si vous avez des questions profitez de discord et de moodle pour les poser à l'enseignant.

0.2 Présentation du projet

Le but de ce projet est que vous implémentiez différents algorithmes vus en cours et que vous manipuliez des appels systèmes. Le projet est divisé en trois parties, les deux premières parties pouvant être programmées dans le langage de votre choix (programmation des algorithmes de pagination et du banquier), et la troisième partie en C++ avec la bibliothèque CImg.

1 Implémentation des algorithmes de pagination $\sim$ 5 points

Dans cette première partie, vous devez programmer dans le langage de votre choix les algorithmes de la seconde chance et l'algorithme optimal pour la pagination. Rappel : ces algorithmes sont utilisés dans la question Q4.4 du TD4. Vous pouvez tester vos deux algorithmes avec l'exemple du TD et comparer vos résultats avec ceux trouvés à la question Q4.4. Votre programme demandera en entrée : le nombre de cadre et le numéro des pages dans leur ordre d'arrivée. Le résultat devra ressembler à la correction de la question Q4.4 en indiquant le nombre de défaut de pages et l'état des cadres à chaque étape. Pensez à préciser le choix effectué par l'algorithme (par exemple, la page P2 est choisie comme victime).
Pour votre structure de données, le choix est libre. Vous pouvez utiliser des tableaux, des files, des listes, des vecteurs, ... Mais essayez d'utiliser des structures que vous maîtrisez pour éviter les erreurs.

2 Algorithme du banquier $\sim$ 5 points

Dans cette seconde partie, vous devez programmer l'algorithme du banquier comme présenté dans votre cours et dans la question Q5.7 du TD5. Pour cet algorithme, nous prendrons en entrée le nombre de types de ressources différentes (4 dans l'exemple du TD), la quantité initiale de ces ressources (3,14,12,12 dans l'exemple du TD) et les matrices Allocations et Maximum. Le reste sera caclulé par votre algorithme. Le résultat attendu doit ressembler à celui obtenu dans le TD et même plus détaillé. Au début de l'algorithme on affiche l'état initiale de toutes les matrices et des vecteurs, puis pour chaque étape on affiche l'état des vecteurs Terminé, DispoTest et DispoTest. On affichera aussi les tests effectués (notamment lorsque l'on teste si les ressources disponibles sont suffisantes pour chaque

processus).
Dans cet exercice vous pouvez utiliser toutes les structures de données que vous connaissez. Pour rappel : une matrice est un tableau à deux dimensions, il est aussi possible d'utiliser des vecteurs pour représenter une matrice.

3 Parallélisation d'un traitement sur les images ∼ 5 points

Proposer un algorithme parallélisable permettant de prendre un ensemble d'images en entrée et d'effectuer un ou l'ensemble des traitements suivants en affichant le nombres d'images par secondes traitées.

- Transformer l'image en noir et blanc
- Redimensioner l'image
- Effectuer un flou

(un bonus de 1 point sera attribué pour chaque algorithme supplémentaire implémenté correctement). Pour réaliser le traitement vous pourrez utiliser l'image `lena.jpg` disponible sur Moodle. On veut alors effectuer le traitement suivant, dupliquer 1000 fois l'image originale avec un appel système, pour avoir 1000 images à traiter dans notre algorithme. Modifier ces 1000 images avec les traitements choisis. Calculer le nombre d'images par secondes traitées. Enregistrer les 1000 nouvelles images traitées. Et écrire dans un fichier le temps nécessaire pour effectuer Le traitement complet. Le programme utilisera les appels systèmes vus en TD.

Prérequis

Pour pouvoir réaliser le projet il faut télécharger la bibliothèque CImg et installer un programme permettant la lecture et l'écriture d'image.
Sur Windows :

- Installation du programme GraphicsMagick pour effectuer la lecture et l'écriture sur des images. `https://sourceforge.net/projects/graphicsmagick/files/graphicsmagick-binaries/1.3.35/`
- Télécharger la bibliothèque CImg à l'adresse suivante : `http://cimg.eu/download.shtml` et placer le fichier CImg.h dans le même répertoire que votre projet.
- Compiler avec les options suivantes : `g++ hello_word.cpp -o hello_word.exe -O2 -lgdi32`

Sur Linux/MacOS :

- Télécharger la bibliothèque CImg à l'adresse suivante : `http://cimg.eu/download.shtml` et placer le fichier CImg.h dans le même répertoire que votre projet.
- Compiler avec les options suivantes :
  - Linux : `g++ hello_world.cpp -o hello_word.exe -O2 -L/usr/X11R6/lib -lm -lpthread -lX11`
  - MacOS : `g++ hello_world.cpp -o hello_word.exe -O2 -lm -lpthread -I/usr/X11R6/include -L/usr/X11R6/lib -lm -lpthread -lX11`

Pour commencer à travailler avec la bibliothèque vous pouvez télécharger le fichier `exempleCImg.cpp`. Dans cet exemple vous avez le chargement d'une image et son affichage. Pour réaliser le projet vous aurez besoin d'accéder aux composantes d'un pixel et de modifier les valeurs d'un pixel avec les fonctions suivantes :

- monImage(x,y,0,0) //récupère la composante rouge du pixel situé aux coordonnées (x,y)
- monImage(x,y,0,1) //récupère la composante verte du pixel situé aux coordonnées (x,y)
- monImage(x,y,0,2) //récupère la composante bleue du pixel situé aux coordonnées (x,y)
- monImage(x,y,0,0)=z //remplace la valeur de la composante rouge du pixel situé aux coordonnées (x,y) par z
- monImage(x,y,0,1)=z //remplace la valeur de la composante verte du pixel situé aux coordonnées (x,y) par z

— monImage(x,y,0,2)=z //remplace la valeur de la composante bleue du pixel situé aux coordonnées (x,y) par z

Attention, la valeur des composantes d'un pixel est codée sur un octet et s'étale donc entre 0 et 255. Le type des composantes est `unsigned char` si vous voulez afficher leur valeur à l'aide d'un entier il faudra utiliser un cast. Exemple avec une variable $c = monImage(x, y, 0, 0)$. Pour afficher c je fais $std :: cout << (int)c << std :: endl$

3.1 Gestion des fichiers

Dans l'exercice 3, il vous est demandé de créer des copies de l'image originale. Pour effectuer cette action vous pouvez regarder du côté des fonctions suivantes :

— `FILE *popen(const char *commande, const char *type);` associée à la fonction : `int pclose(FILE *stream);` provenant de la bibliothèque `unistd.h`.

— `system(const char*)` provenant de la bibliothèque `stdlib`.

— `std::ifstream src(const char*, std::ios::binary);`
`std::ofstream src(const char*, std::ios::binary);`
provenant de la bibliothèque `fstream`

Il est possible de parcourir un répertoire en utilisant la structure de donnée dirent de la bibliothèques `dirent.h`. Le code suivant permet d'afficher les éléments du répertoire courant sur windows :

```
1  DIR *dir;
2  struct dirent *ent;
3  if ((dir = opendir (".")) != NULL) {
4      print all the files and directories within directory
5      while ((ent = readdir (dir)) != NULL) {
6          printf ("%s\n", ent->d_name);
7      }
8      closedir (dir);
9  } else {
10     could not open directory
11     perror ("");
12     return EXIT_FAILURE;
13 }
```