Table des matières

Projet programmation système W05

Introduction

L'objectif de la semaine est d'implémenter deux fonctionnalités pour le gestionnaire d'images :

— la commande `help`, élément standard et indispensable de toute utilitaire de commande ;
— la commande `create`, pour créer un nouveau fichier (vide) au format `imgStore` (= une nouvelle base d'images).

Le but de cet exercice est d'apprendre à écrire des structures de données sur disque en utilisant les opérations de base d'entrées/sorties.

Comme les semaines précédentes, il s'agira d'écrire votre code en modifiant des éléments fournis.

Afin de simplifier l'exercice, un certain nombres de paramètres, notamment le nombre maximal d'images ainsi que les résolutions « *thumbnail* » et « *small* », sont pour le moment fixées par des constantes à l'intérieur du code fourni (cf la fonction `do_create_cmd()` dans `imgStoreMgr.c`), plutôt que des arguments passés sur la ligne de commande ; ce qui sera le cas plus tard dans le projet.

Matériel fourni

Nous vous fournissons cette semaine du nouveau matériel dans votre dépôt GitHub, au travers d'un nouveau répertoire nommé `provided`. Afin d'éviter tout problème, **le contenu de ce répetoire `provided` ne devra jamais être modifié !**

Pour le récupérer :

1. commencez par sauvegarder et publier (`git commit`) vos modifications en cours ;
2. mettez à jour votre dépôt à l'aide de

```
git pull
```

3. quand vous souhaitez travailler sur le nouveau matériel fourni, commencez par le recopier dans votre répertoire `done/` et enregistrez le dans git (`git add`). Nous donnons un exemple complet ci-dessous.

Pour cette semaine, nous vous fournissions (dans `provided`, donc) le nouveau fichier suivant :

— `imgst_create.c` : implémentation de la commande `create`.

Pour l'ajouter à votre projet, faites donc (depuis `done`) :

```
git pull
cp ../provided/imgst_create.c .
git add ./imgst_create.c
git commit -m "Importation de nouveau matériel"
git push
```

puis travaillez normalement sur ce nouveau fichier **dans votre répertoire `done`**. Nous répétons ici que le répertoire `provided` ne doit en aucune façon être modifié par vous-mêmes.

Pour le reste, il vous faudra continuer à modifier sur les fichiers déjà utilisés la semaine passée : `imgStoreMgr.c`, `imgStore.h` et bien sûr le `Makefile`.

Travail à faire

Pour l'exercice de cette semaine, vous utiliserez simplement la structure de type `imgst_file` de la semaine précédente, définie pour 10 (`MAX_MAX_FILES`) images au maximum. En conséquence, votre implémentation ne créera (pour le moment) que des base d'images de cette taille.

Votre travail de la semaine consiste en cinq modifications, résumées ici et détaillées plus bas si nécessaire :

1. prototyper la fonction `do_create()` dans le fichier `imgStore.h` ; cette fonction doit prendre deux arguments : le nom du fichier et une structure de type `imgst_file` ; elle retourne un code d'erreur (`int`) ou `0` s'il n'y a pas d'erreur ;
2. dans le fichier `imgst_create.c`, implémenter la fonction `do_create()` ayant pour but de créer une nouvelle base d'images dans un fichier (binaire) sur disque ;
3. compléter la fonction `do_create_cmd` dans le fichier `imgStoreMgr.c` afin d'appeler correctement `do_create` ;
4. définir la fonction `help()` qui imprimera les instructions pour utiliser l'utilitaire `imgStoreMgr` ;
5. mettre à jour le `Makefile` pour la compilation et création de l'utilitiaire `imgStoreMgr`.

Définir `do_create()`

Cette fonction doit terminer d'initialiser la structure `imgst_file` reçue avant de l'écrire sur le disque, le « *header* » en premier, puis les « *metadatas* ». Elle doit utiliser les fonctions standard C d'entrée/sortie pour créer la nouvelle base d'images dans un fichier *binaire* sur disque. Si le fichier existe déjà, on l'écrase simplement (sans erreur).

Il est important de bien initialiser explicitement tous éléments *pertinents* avant l'écriture. Et il est évidemment essentiel d'écrire le tableau de `metadata` de la bonne taille dans le fichier.

Il est également important de bien traiter tous les cas d'erreurs possibles. En absence d'erreur, `do_create()` doit retourner 0 ; en cas d'erreur, retourner le code de valeur correspondant tel que défini dans `error.h`.

Cette commande n'étant utilisée qu'une fois (pour créer une base) et toujours depuis l'utilitaire ligne de commande `imgStoreMgr` (ne sera jamais lancée depuis un serveur Web, par exemple), nous allons **exceptionnellement** y ajouter un effet de bord sous la forme d'un affichage indiquant le (*vrai*) nombre d'objets sauvés sur le disque.
Dans le cas présent (un « *header* » puis dix « *metadatas* »), on aura alors l'affichage suivant :

```
11 item(s) written
```

`11` parce que le « *header* » puis chacun des dix « *metadatas* » ont été écrits avec succès par `fwrite`.

Compléter `do_create_cmd()`

Nous vous fournissons une implémentation incomplète de `do_create_cmd()`. Dans le cadre de votre solution, vous devez créer un `imgst_file`, initialiser les champs `max_files` et `res_resized` de son « *header* » avec les valeurs fournies, puis appeler ensuite `do_create` (qui initialisera les autres champs).

`do_create_cmd()` doit retourner le code d'erreur retourné par `do_create()`.

Définir `help()`

La commande `help` est destinée à être utilisée dans deux cas différents (déjà traités) :

1. lorsque les arguments passés à l'utilitaire ne sont pas valables ;
2. lorsque l'utilisateur demande explicitement la liste des possibilités en tapant `imgStoreMgr help`.

L'output de la commande doit avoir *exactement* le format suivant :

```
imgStoreMgr [COMMAND] [ARGUMENTS]
```

```
help: displays this help.
list <imgstore_filename>: list imgStore content.
create <imgstore_filename>: create a new imgStore.
```

Pour résoudre cette partie (facile), écrire la fonction à l'endroit prévu dans `imgStoreMgr.c`.

Tests

Testez vos deux nouvelles commandes (utilisez `help` pour savoir comment utiliser `create` ;-P).

Pour vérifier que le fichier binaire a été correctement écrit sur disque, utilisez la commande `list` de la semaine passée.

Nous vous fournissons aussi dans le répertoire `provided/tests` deux Shell scripts qui font automatiquement des tests pour le travail de cette semaine. Ces deux tests sont ceux que nous tournons dans `make feedback`. Pour les faire tourner en local (largement recommandé **avant** de faire des `make feedback` !), faites :

```
make check
```

Et donc, comme d'habitude, nous fournissons également, *à bien plaire*, un `make feedback` (`make feedback-VM-CO` si vous travaillez sur les VM de l'Ecole) qui donne un retour *partiel* sur votre travail. Ceci est normalement à utiliser pour une vérification *minimale* **finale** de votre travail, avant de rendre. Préférez auparavant faire des tests **locaux** directement sur votre machine (et y compris plus de tests que vous aurez vous-même ajouté si nécessaire).

L'image Docker utilisé par `make feedback` est chaque semaine marquée de l'étiquette `latest`, mais si vous souhaitez faire tourner le feedback d'une semaine spécifique, changez (dans le `Makefile` à la ligne qui définit `IMAGE`) cette étiquette `latest` par `weekNN` où `NN` est le numéro de semaine désiré, p.ex. :

```
IMAGE=chappeli/pps21-feedback:week04
```

pour le feedback de la semaine passée *uniquement*, ou :

```
IMAGE=chappeli/pps21-feedback:week05
```

pour le feedback de cette semaine (lequel inclut aussi celui de la week04), lorsque vous voudrez le lancer dans une future semaine (p.ex. si vous avez pris du retard — ce qui est déconseillé).

Rendu

Comme pour la semaine passée, vous n'avez pas à rendre de suite le travail de cette semaine de projet. Celui-ci ne sera à rendre qu'à la fin de la semaine 8 (délai : le dimanche 25 avril 23:59) en même tant que le travail des semaines 4 à 7.

Nous vous conseillons cependant toujours de travailler régulièrement et faire systématiquement ces commits réguliers, au moins hebdomadaires, lorsque votre travail est opérationnel. Cela vous aidera à sauvegarder votre travail et à mesurer votre progression.

Pour sauvegarder la partie correspondant à cette semaine, ajoutez les nouvelles versions des fichiers `imgStoreMgr.c`, `imgStore.h` et `Makefile`, ainsi que le nouveau fichier `imgst_create.c` dans le répertoire `done/` de votre GitHub, puis « pousser » le résultat vers GitHub (`commit` plus `push`).