TP9 et TP10 : Mini S.G.F.

CE TP DOIT ÊTRE RENDU.
Les modalités seront précisées la semaine prochaine.

1 Programmation d'un mini S.G.F.

Nous allons, pendant deux séances de travaux pratiques, reconstruire les fonctions d'un S.G.F. très simplifié.

- Commencez par récupérer les sources de ce mini SGF.
- Le projet est organisé de la manière suivante :
  - « **sgf-disk.c** » : entrée/sortie de blocs vers le disque simulé par le fichier `disk`.
  - « **sgf-fat.c** » : gestion de la FAT en mémoire et sur disque.
  - « **sgf-dir.c** » : lecture, écriture et effacement des couples (nom,adr-i-node) dans le répertoire.
  - « **sgf-io.c** » : opérations d'ouverture, de lecture, d'écriture et de fermeture des fichiers.

Le module « **sgf-fat.c** » est complet. Les fonctions à programmer se trouvent dans « **sgf-dir.c** » et « **sgf-io.c** ».

2 Lecture dans un fichier

Il est fortement conseillé de programmer les fonctions en respectant l'ordre suivant :

1. Programmez la fonction « *list_directory* » du fichier « **sgf-dir.c** » et utilisez la dans « **main.c** » pour afficher le contenu du disque. Vous pouvez utiliser la commande *format* livrée avec le projet pour fabriquer un disque vierge avec un fichier de test placé à l'intérieur.
2. Programmez la fonction « *sgf_read_bloc* » du fichier « **sgf-io.c** » et testez la dans « **main.c** »en lisant le contenu d'un fichier.

3 Accès direct en lecture

On vous demande de réaliser la fonction « seek » qui réalise le déplacement du pointeur ptr en lecture. Pour ce faire, suivez les étapes ci-dessous :

1. Ajoutez au fichier « sgf-io.h » la déclaration suivante :

```
int sgf_seek (OFILE* f, int pos);
```

2. Ajoutez l'implantation de cette fonction dans le fichier « sgf-io.c ». Cette fonction va déplacer le pointeur associé au fichier ouvert en utilisant le deuxième paramètre. Si la position demandée n'est pas un multiple de la taille des blocs, alors le bloc en question devra être chargé dans le tampon. **Attention** : prenez soin de limiter la lecture de bloc notamment quand la nouvelle position concerne le même bloc que la position précédente. Cette fonction renvoie 0 en cas de succès et -1 sinon.

3. Testez cette fonction en lisant, dans un fichier, les caractères dont la position est multiple de 8.

4 Écrire dans un fichier caractère après caractère

Le principe est le même que la lecture. Les caractères sont placés dans le « *buffer* » tant que celui-ci n'est pas plein. Une fois le « *buffer* » complété, un nouveau bloc est ajouté au fichier. Pour re-programmer les fonctions d'écriture, suivez les étapes ci-dessous :

1. Re-programmez la fonction « *sgf_putc* » du fichier « **sgf-io.c** ».

2. Terminez la fonction « *sgf_append_block* » et testez les deux routines. **Attention** : normalement, à cette étape, vous devez perdre les derniers caractères du fichier puisque la fonction de fermeture n'est pas terminée. Vous pouvez utiliser le petit code ci-dessous pour tester les fonctions d'écriture.

```
OFILE* file = sgf_open("essai.txt", WRITE_MODE);
sgf_puts(file, "Ceci est un petit texte qui occupe \n");
sgf_puts(file, "quelques blocs sur ce disque fictif. \n");
sgf_puts(file, "Le bloc faisant 128 octets, il faut \n");
sgf_puts(file, "que je remplisse pour utiliser \n");
sgf_puts(file, "plusieurs blocs.\n");
sgf_close(file);
```

3. Re-programmez la fonction de fermeture « *sgf_close* ».
4. Terminez par la fonction de destruction « *sgf_remove* ». Faites en sorte que cette fonction imprime l'état du disque (nombre de blocs libre par exemple).

5 Ouvrir un fichier en mode ajout

On vous demande de mettre en place un mode d'ouverture en ajout que ne vide pas le fichier mais qui permet de l'agrandir. Suivez les étapes ci-dessous :

1. Ajoutez au fichier « **sgf-io.h** » la déclaration suivante

```
#define APPEND_MODE (2)
```

2. Modifiez la fonction « *sgf_open* » afin qu'elle traite cette nouvelle valeur en appelant la fonction « *sgf_open_append* ».

3. Commencez par **recopier** la fonction « *sgf_open_read* » pour créer « *sgf_open_append* ». Cette dernière doit maintenant lire le dernier bloc du fichier si la taille n'est pas un multiple de la taille des blocs. (il faut lire le dernier bloc incomplet pour le compléter).

4. **Attention** : la fonction « *sgf_append_block* » doit maintenant être modifiée. En mode ajout, il ne faut pas ajouter un nouveau bloc mais simplement récrire la **nouvelle version du dernier bloc**. Une fois cette opération réalisée, **nous ne sommes plus en mode ajout**.

5. Testez ce mode en vidant un fichier (ouverture en écriture) puis en l'ouvrant 500 fois en mode ajout afin, à chaque fois, d'ajouter un seul caractère.

6 Écrire une zone de données

On vous demande d'ajouter la fonction classique d'écriture d'une zone de données :

1. ajoutez au fichier « **sgf-io.h** » la déclaration suivante

```
int sgf_write (OFILE* f, char *data, int size);
```

2. programmez cette fonction sans utiliser la fonction « *sgf_putc* ». L'idée est de compléter le bloc courant, de l'écrire et de recommencer tant que la zone n'est pas entièrement écrite.