Manipulations de fichiers

Exercice 1. Questions de cours Les questions de cours sont à destinées à vous permettre de vérifier votre compréhension du cours. Elles sont à travailler à l'avance et ne seront pas traitées en TD ou TP.

1. Pourquoi faut-il ouvrir les fichiers ?
2. Donnez trois modes de contrôle d'accès différents.
3. Qu'est-ce qu'un File Control Bloc ?
4. Quels sont les avantages et inconvénients des allocations contiguë, chaînée et indexée ?

Exercice 2. Copie de fichiers L'objectif de cet exercice est de programmer un équivalent simplifié de la commande de copie de fichier `cp`. Chaque partie demande d'étendre le programme précédent afin d'ajouter des fonctionnalités. Assurez-vous de bien tester votre code à chaque étape.

La documentation de toutes les fonctions à utiliser est disponible en section 2 des pages de manuels. Par exemple pour obtenir la documentation de la fonction `stat` il vous suffit d'utiliser la commande `man 2 stat`. Attention, les pages de manuel vous indiquent les fichier d'en-tête à inclure afin de pouvoir utiliser les fonctions qu'elles décrivent. N'oubliez pas d'inclure ces fichiers au début de votre programme.

1. Étape 1 : Vous commencerez par écrire un premier programme de copie de fichier simple qui copie un seul fichier. Ce programme prend en paramètre un fichier source et un fichier cible et réalise la copie.

Vous aurez besoin des fonctions `open`, `close`, `read` et `write`. Il vous est conseillé d'implémenter la copie dans une fonction séparée afin de faciliter les étapes suivantes.

La copie doit être réalisée par blocs. Il s'aggit de lire un bloc de donnée depuis le fichier source et de l'écrire dans le fichier cible et de répéter jusqu'à ce que la fin du fichier source soit atteinte.

2. Étape 2 : Votre premier programme ne préserve pas les droits d'accès aux fichiers qu'il copie. Vous allez maintenant le modifier pour que lors de la copie il récupère les droits du fichier source et les appliques au fichier cible.

Vous aurez besoin des fonctions `stat` et `chmod`, ou `fstat` et `fchmod`. Vous devez maintenant avoir une fonction qui réalise correctement la copie d'un fichier de manière fiable.

3. Étape 3 : Nous allons maintenant ajouter la gestion de la copie de répertoires. Dans un premier temps nous considérerons le cas simple de la copie d'un répertoire ne contenant que des fichiers standards, pas de sous répertoire. Modifiez votre programme de manière à ce qu'il copie tous les fichiers contenus dans un répertoire source vers un répertoire cible déjà existant.

Vous aurez besoin des fonctions `opendir`, `closedir` et `readdir`. (Attention, il n'est pas garanti que les noms des répertoires source et destination finissent par le caractère '/'.

4. Étape 4 : La dernière étape de la réalisation de ce programme de copie consiste à rendre la copie récursive. Votre programme final doit être capable de copier aussi bien des fichiers que des répertoires, et pour ces derniers la copie doit être récursive.

Les fonctions `stat` et `fstat` permettent de différentier les fichier des répertoires. Vous aurez besoin de la fonction `mkdir`.

Exercice 3. Performances Sur un petit nombre de petits fichiers, les performances de votre programme sont à la fois peu importantes et difficiles à mesurer. Un cas où les performances peuvent être mauvaises est la copie d'un grand nombre de petits fichiers. Dans ce cas il est difficile de régler le problème car la copie d'un grand nombre de fichiers nécessitera toujours un grand nombre d'appels système.

Le cas de la copie de gros fichier est plus intéressant car la manière de copier à une grande importance. Afin de tester les performance de votre programme, il va falloir que vous mesuriez sont temps d'exécution lors de la copie de gros fichiers. Si vous ne disposez pas de tels fichier vous pouvez en créer avec un contenu aléatoire à l'aide de la commande suivante :

```
dd if=/dev/random of=toto count=2048
```

Le paramètre `of` permet de spécifier le nom du fichier à créer et le paramètre `count` indique sa taille en multiple de 512o. La commande précédente va donc créer un fichier de 1Mo.

1. Étudier l'impact de la taille du bloc utilisé pour la copie sur les performance de la copie.

Choisir une bonne taille de bloc permet de réaliser la copie de manière relativement rapide et portable. Toutefois, un grand nombre d'appels système sont quand même nécessaires et les performances restent en dessous de ce que le materiel peut fournir. La plupart des systèmes ont maintenant ajouté des appels système spéciquement destinés à réduire ce problème. Sous Linux, c'est le rôle de l'appel système `copy_file_range` qui est destiné à remplacer la boucle de copie par blocs. L'utilisateur est toujours responsable d'ouvrir et fermer les différents fichiers mais la copie des données d'un fichier à l'autre est directement réalisée par cet appel. Cet appel n'étant pas standard, il est nécessaire d'ajouter la ligne :

```
#define _GNU_SOURCE
```

avant l'inclusion des fichiers d'en-tête dans votre programme pour qu'il soit disponible.

2. Modifiez votre fonction de copie de manière à ce qu'elle effectue la copie à l'aide de ce nouvel appel système et évaluez ses performances.