UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 3 Informatique
Info0601 – 2020/2021
***Resp. :* Cyril Rabat**

Travaux dirigés n° 1

Fichiers et systèmes de fichiers

Exercice 1 (Table d'adresses)
1°) Nous supposons qu'un système de fichiers est basé sur la notion de table d'adresses (comme un *i-node*) pour mémoriser les blocs utilisés pour un fichier.

a) Sans niveau d'indirection et en considérant qu'une table possède un nombre fixe k d'enregistrements, quelle est la taille maximale d'un fichier ?

b) Pourquoi est-il préférable que la table d'adresses possède un nombre d'enregistrements maximum ?

2°) Dans les *i-nodes* du système *ext2*, les tables d'adresses possèdent 15 enregistrements, les 3 derniers correspondant à des niveaux d'indirection différents. Les adresses de bloc ont une taille de 4 octets.

a) Rappelez le principe de l'indirection. Illustrez votre réponse par un schéma.

b) Avec des blocs de 512o et en supposant que les 3 derniers enregistrements des *i-nodes* permettent un seul niveau d'indirection, quelle est la taille maximale d'un fichier ?

c) En réalité, le treizième enregistrement possède un niveau d'indirection, le quatorzième en possède deux et le quinzième en possède trois. Quelle est la taille maximale d'un fichier ?

d) Pourtant *ext2*, qui est basé sur ce principe, permet des tailles de fichier de 2TiB maximum. Comment l'expliquer ?

e) Pour un fichier qui possède la taille la plus grande possible (calculée à la question 2.c), combien de tables d'adresses sont nécessaires ? Quelle est la taille de l'ensemble de ces tables ?

Exercice 2 (Fichier à trous)
Nous souhaitons réaliser une application permettant d'enregistrer des contacts dans un fichier. Un contact contient un nom, un prénom et une adresse de courriel.

1°) Nous supposons que chaque champ d'un contact possède une taille fixée.

a) Proposez une structure en C pour représenter un contact (nommée `contact_t`).

b) Expliquez comment ajouter un contact dans le fichier (fonction *ajout*), afficher tous les contacts (fonction *affichage*) et chercher un contact en fonction d'un nom (fonction *recherche*).

c) Expliquez quels sont les problèmes rencontrés pour la suppression et donnez une solution qui n'impose pas le déplacement de contacts dans le fichier.

d) Y-a-t-il des modifications à apporter pour les fonctions présentées à la question 1.b ?

2°) Nous supposons maintenant que les champs d'un contact possèdent des tailles variables.

a) Rappelez le problème de la sauvegarde d'une chaîne de caractères dans un fichier et de sa lecture.

b) Donnez la nouvelle structure en C d'un contact (nommée `contactD_t`).

c) Donnez le code de la procédure `void ajouter(int fd ,contactD_t contact)` permettant de sauvegarder un contact dans un fichier dont `fd` est le descripteur associé. Le contact est sauvegardé à la position en cours.

d) Donnez maintenant le code de la fonction `contactD_t lire(int fd)` permettant de lire le contact situé à la position en cours dans le fichier.

e) Que se passe-t-il lors d'une suppression d'un contact dans le fichier ? Qu'est-ce que cela change pour les différentes fonctions présentées à la question 1.b ?

f) Proposez une solution pour limiter les modifications et précisez les limitations de votre solution.

3°) Nous utilisons une table de positions qui permet de mémoriser la position dans le fichier de chaque contact. Elle est sauvegardée au début du fichier, avant les contacts.

a) Pourquoi est-il plus simple d'utiliser un nombre fixe d'enregistrements dans cette table ?

b) Que se passe-t-il si le nombre de contacts dépasse ce maximum ? Proposez une solution permettant d'ajouter autant de contacts que souhaité.

c) Donnez la structure en C de la table de positions et donnez le code en C permettant de la sauvegarder dans un fichier.

d) Expliquez comment peut-on supprimer un contact dans le fichier, en sachant que la position doit être conservée pour l'ajout de futurs enregistrements. Vous pouvez modifier la structure de la table.

e) Expliquez comment ajouter un contact dans le fichier, en sachant qu'un contact peut être ajouté soit dans un trou, soit à la fin du fichier.

4°) Pour simplifier la gestion des trous dans notre fichier, nous exploitons une autre structure que nous appelons la table de vide. Celle-ci permet de mémoriser la position et la taille de chaque trou. Elle fonctionne sur le même principe que la table des positions et elle est sauvegardée après celle-ci dans le fichier.

a) Donnez la structure en C de cette table.

b) Expliquez comment ajouter un contact dans le fichier.

c) Expliquez comment supprimer un contact dans le fichier.

5°) Pour améliorer notre application, nous souhaitons pouvoir associer plusieurs adresses de courriel à un contact. Donnez toutes les structures et détaillez toutes les fonctions.