Portail IE / INF2 25 mars 2019
Contrôle continu TP 30 mn

1. Que signifie SGF ?

Système de gestion de fichiers.

Attention : Certains ont tenté le coup de Smart game format ou des mélanges bizarres, genre System Gestion File !

2. Quelle est le rôle des nœuds ?

Les nœuds sont les entrées du cache. Ils représentent des contenus de fichier qui ont déjà été lus (les `DNODE`) ou des répertoires qui ont été ouverts.

3. Quelle fonction de `fs.py` permet de passer de nœud à nom, et quelle autre de passer de nom à nœud ?

`NODE` → nom : `fs_lookup_in_dcache`
nom → `NODE` : `name_of_node`

4. À quoi sert le cache de données ?

Le cache de données sert à accélérer l'accès au disque en stockant des contenus de fichiers qui ont une grande probabilité d'être relus ou écrits prochainement afin de ne pas avoir à les relire sur le disque, et donc permettre d'y accéder à la vitesse de la mémoire et pas à celle du disque. Pour cela, il est structuré en pages qui correspondent à des secteurs du disque.

Attention : le cache de données ne peut généralement pas contenir toutes les données du disque ; il y a beaucoup moins de pages disponibles que de secteurs. Il faut donc choisir quoi y mettre, et le principe de localité dit que ce sont les contenus des fichiers les plus récemment lus ou écrits. En suivant ce principe, à chaque instant, il contiendra la partie du disque qui semble la plus utile immédiatement.

Attention : on a vu en cours que la notion de « plus récemment lus ou écrits » n'est pas réalisée en stockant des dates d'accès ni en stockant des fréquences d'utilisation, mais juste en utilisant un bit *used* par page de cache.

Attention : il est vraiment très insuffisant de dire que le cache sert à stocker des données ; tellement d'autres choses servent à stocker des données ! Dans le même ordre d'idée, il est bizarre de dire qu'il sert à désynchroniser ; personne n'a envie de désynchroniser ! La désynchronisation est un prix temporaire à payer, pas un service. Et il est faux de dire que le cache sert à éviter l'usure du disque. C'est avec les disques SSD (semi-conducteur ou *flash*) que se pose le problème de l'usure du disque.

5. Comment le TP rend-il compte de l'effet du cache de données ?

Il est géré par la bibliothèque `fs.py`. Tous les accès au disque simulé par la bibliothèque sont réalisés via un cache, voir les fonctions `fs_..._in_dcache`. Le simulateur du disque tient à jour les scores de réussite de l'accès au disque via le cache : *cache hit* et *cache miss*. La fonction `SYNC` resynchronise le cache et le disque simulé. La fonction `DUMP` permet de voir l'état du cache et de ses bits *used* et *dirty*. Enfin, les scénarios de la dernière question permettent d'observer l'effet du cache et de sa taille sur les performances globales.

Attention : il est faux de dire que la gestion du cache par le simulateur se traduit par une exécution plus rapide de vos programmes :

- Le disque est simulé ; il n'a donc pas les temps d'accès d'un vrai.
- Les données de chaque fichier sont supposées tenir sur un secteur ; du coup l'accès est largement facilité.
- Les expériences envisagées contiennent très peu de données ; en tout cas rien qui pourrait stresser l'exécution de vos programmes.

On souhaite développer une nouvelle fonction `CP` permettant la copie de fichier. Elle devra répondre à la spécification suivante :
`CP` prend deux paramètres qui sont des noms, et crée un nouveau fichier qui a pour contenu celui du premier nom, et pour nom, le second nom.

```
;; Copy a file
;; CP : string * string -> proc
```

6. Lister les propriétés qu'il faudrait vérifier pour qu'une copie soit possible.

Il faut s'assurer :

1. que le nom de l'original est bien celui d'un fichier, et pas d'un répertoire, ou pire, de rien du tout,
2. que le nom de la copie n'est pas déjà utilisé,
3. et qu'il reste de la place dans le disque pour créer le fichier copie.

Attention : même vide, un fichier a un contenu, comme un ensemble vide est tout ce qu'il y a de plus normal. Ce n'est pas quelque chose à vérifier. On peut copier un fichier vide ; cela crée un autre fichier vide.

Attention : beaucoup parlent de droit d'accès, d'autres de formats de fichier. Mais est-ce que le TP parlait de cela ? Ça ne sert à rien d'étaler sa science quand ça ne répond pas à la question.

Attention : ce n'est pas l'endroit pour dire comment faire la vérification (question 7), et encore moins l'endroit pour dire comment faire la copie (question 8).

7. Pour l'une de ces propriétés (selon votre choix) dire comment la vérifier à l'aide des fonctions `fs_...`

Pour chaque propriété ci-dessus, voici une réponse possible :

1. `fs_lookup_in_dcache` : vérifier que le résultat n'est pas `False` et que `is_FNODE` répond `True` ;

2. `fs_lookup_in_dcache` : vérifier que le résultat est `False` ;
3. `fs_getfreeblock` : vérifier que le résultat n'est pas `False`.

Attention : quand vous avez un choix comme ici, dites bien lequel vous faites. Ce n'est pas au correcteur d'essayer de le deviner ! C'est particulièrement important ici où la fonction `fs_lookup_in_dcache` est utilisée de deux façons opposées par deux conditions différentes.

8. En supposant ces propriétés satisfaites a priori dire comment réaliser la copie à l'aide des fonctions `fs_...`

On doit

- lire le fichier original : `fs_read_in_dcache` ;
- puis créer un nouveau fichier : `fs_touch_in_dcache` ;
- puis y inscrire le contenu original : `fs_write_in_dcache` .

Attention : on peut y voir la mise bout à bout de `READ`, `TOUCH` et `WRITE`. Cependant, on a là un système en couche : avec une couche système constituée des `fs_...` et une couche utilisateur avec les `CD`, `LS`, … Donc, pour réaliser `CP`, il vaut mieux utiliser les `fs_...` que les `READ`, etc. Par exemple, `TOUCH` marche aussi quand le fichier à créer existe déjà, alors que pour `CP` on ne veut vraiment pas qu'il existe déjà. En fait, la fonction `TOUCH` ne peut absolument pas être utilisée pour tester si quelque chose existe, car si la chose n'existe pas, elle la crée ! Plus généralement, les fonctions `CD`, `LS`, …, sont faites pour afficher (`print`) des choses sur le terminal de l'utilisateur, alors que pour les réaliser on a besoin de fonctions qui retournent (`return`) des résultats.

9. Proposer un scénario de test pour la fonction `CP`.

Il faut s'assurer de plusieurs choses :

- Que le contenu de la copie est égal à celui de l'original.
- Mais qu'il s'agit bien d'une copie, c-à-d. qu'on peut modifier la copie sans altérer l'original, et modifier l'original sans altérer la copie.
- Mais aussi vérifier les cas de la question 6.

D'où par exemple :

```
TOUCH("from") ; WRITE("from", "OLD")
CP("from", "to1") ; CP("to1", "to2")
LS()
print( READ("from") == READ("to1") )
print( READ("to1") == READ("to2") )
WRITE("to1", "NEW")
print( READ("from") == "OLD" )
print( READ("to1") == "NEW" )
print( READ("to2") == "OLD" )
CP("from", "to1") ; CP("nofile","to")
```

Le test de ce que la vérification qu'il y a assez de place dans le disque pour effectuer la copie est bien faite est plus délicat, car cela suppose de mettre le disque

exactement dans la situation où il n'a plus de place. Pourtant les systèmes grand public doivent aussi être testés à ce niveau de détail.

On peut aussi tester à un plus bas niveau en utilisant `DUMP` pour aller voir directement le résultat sur disque. Mais cette méthode ne marchera pas pour tester les violations des conditions de la question 6.

Attention : certains ont bien pensé à tester la vérification des conditions de la question 6, mais uniquement à ça. Dans ce cas, une fonction `CP` qui vérifie ses paramètres et l'état du système, mais n'en fait rien passerait le test ! Méfiez-vous des fonctions qui ne font rien ; elles passent beaucoup de test !

10. Donner rapidement l'idée de ce qu'il faudrait faire pour copier des répertoires.

Il faudrait créer un nouveau répertoire qui aurait le second nom, et répéter les opérations ci-dessus pour tous les habitants du répertoire d'origine, et quand un habitant est un sous-répertoire, il faudrait en créer une copie et recommencer pour ses habitants, et ainsi de suite récursivement. Les habitants et sous-habitants sont copiés d'un répertoire à l'autre en conservant leurs noms.

Le moyen de connaître les habitants est `fs_list_in_dcache`, celui de créer de nouveaux répertoires est `fs_mkdir_in_dcache` et celui de passer de répertoire en répertoire est `fs_cd_in_dcache`.

On peut voir l'ensemble comme une combinaison de `LS`, `CD`, `MKDIR`, `READ`, `TOUCH` et `WRITE`, avec la réserve exprimée à la question 8.

Attention : ce n'est pas une super idée de réutiliser la fonction `CP` car celle-ci suppose que le nom de l'original et le nom de la copie soit différents ; normal ils habitent le même répertoire. Mais ici, la copie des habitants du répertoire d'origine se fait dans un autre répertoire mais avec le même nom pour les habitants copiés. Cette copie-là n'obéit donc pas aux mêmes contraintes que `CP`.

Bilan :

Remarque très générale : en réponse à une question, montrez d'abord que vous avez compris la question, et ensuite que vous savez y répondre. C'est fondamental.

À des questions du genre « À quoi sert X ? », veillez bien à répondre à quoi ça sert plutôt que ce que ça fait ou comment c'est fait. Dans « À quoi ça sert ? » pensez service. Une bonne façon de mieux y arriver est de reprendre le verbe dans la réponse : « X sert à … ». C'est un principe très général qui s'applique à toute sorte de questions. Même si vous n'écrivez pas la réponse comme ça, pensez-là comme ça.

Attention : j'ai bien spécifié en cours que je n'attendais pas du code. Ce qui m'importe ce sont vos idées, pas votre code. Évidemment, je n'ai pas sanctionné ceux qui exprimaient leurs idées en code, même si c'était pour eux une dépense

d'énergie inutile. Mais dans certains cas, le code était tellement incompréhensible que je n'ai rien pu en tirer.

Attention : il y a eu énormément de questions laissées sans réponse, et pourtant, le jour du contrôle vous avez été très nombreux à partir bien avant la fin. Vous devez vraiment utiliser tout le temps disponible pour répondre au mieux. Je peux facilement interpréter charitablement une demi-réponse, mais je ne peux rien faire avec une non-réponse.

Attention : ma remarque sur l'orthographe et la grammaire tient toujours. Certaines copies sont vraiment incompréhensibles. C'est à vous d'agir.