EFREI - Ing1 **2002**
Systèmes d'Exploitation
TD 4

Système de gestion de fichiers

---

1 – Soit un système de fichiers Unix dans lequel un bloc disque a une taille de 4 Ko et un pointeur vers un bloc disque est représenté sur 4 octets.
a - Quelle est la plus grande taille possible d'un fichier ?
b – Si le système d'exploitation a déjà chargé en mémoire vive l'inode d'un fichier, quel est le nombre d'accès disque nécessaires pour charger le bloc de données numéro 600 en mémoire ?

a – L'ancien système Unix (vesrion BSD 4.1) a un inode qui a :
- 10 pointeurs directs vers des blocs de données (10 blocs)
- 1 pointeur en simple indirection vers un bloc qui contient des pointeurs vers 1024 blocs de données supplémentaires (1024 blocs)
*- 1 pointeur en double indirection vers un bloc qui contient des pointeurs vers 1024 blocs dont chacun contient des pointeurs vers 1024 blocs supplémentaires de données (1024*1024 blocs)*
*- 1 pointeur en triple indirection vers un bloc qui contient des pointeurs vers 1024 blocs dont chacun contient des pointeurs vers 1024 blocs dont chacun contient des pointeurs vers 1024 blocs supplémentaires de données (1024*1024*1024 blocs)*
*Nombre total de blocs = 10 + 1024 + 1024*1024 + 1024*1024*1024 = 1 073 741 834*
Taille maximale d'un fichier** = 1 073 741 834 * 4096 = 4 398 046 552 064 octets = **4 To

*b – **2 accès en lecture sont nécessaires***
- lecture d'un bloc indirect en simple indirection (blocs 11 à 1034)
- lecture du bloc de données pour le bloc 600

2 – Quand le fichier */users/efrei/ing1/td4* est ouvert, plusieurs accès disque sont nécessaires pour la lecture des blocs de l'inode et du répertoire. Si l'inode du répertoire racine est toujours en mémoire et que la taille d'un répertoire est égale un bloc, calculer le nombre d'accès disque nécessaires pour ce fichier.

L'accès à l'inode du répertoire racine ne requiert pas d'accès disque, on a donc :
- *lecture du répertoire / pour chercher users*
- *lecture de l'inode pour /users*
- *lecture du répertoire /users pour chercher efrei*
- *lecture de l'inode pour /users/efrei*
- *lecture du répertoire /users/efrei pour chercher ing1*

– *lecture de l'inode pour /users/efrei/ing1*
– *lecture du répertoire /users/efrei/ing1 pour chercher td4*
– *lecture de l'inode pour /users/efrei/ing1/td4*

*Au total, il faut **8 accès disque** avant que l'inode du fichier ne soit en mémoire*

3 – Un éditeur de texte a segment de code de taille 100 Ko, un segment de données initialisées de 30 Ko et de données non initialisées *BSS (Block Started by Symbol)* de 50 Ko. La taille initiale de la pile est de 10 Ko. On lance simultanément 3 exemplaires de cet éditeur.
a – Calculer la quantité de mémoire nécessaire si le code de l'éditeur est partagé
b - Calculer la quantité de mémoire nécessaire si le code de l'éditeur n'est pas partagé

*a – Avec le code partagé, on a 100 Ko pour le code et chacun des processus a besoin de 80 Ko pour son segment de données et 10 Ko pour sa pile, ce qui donne un total de **370 Ko** de mémoire*
*b – sans code partagé, chaque processus a besoin de 190 Ko, ce qui donne un total de **570 Ko** de mémoire*

4 – Un éditeur de texte sous Unix sauvegarde un nouveau fichier sur le disque. Le fichier a une taille de 15678 octets et le bloc disque a une taille de 1 Ko.
Unix écrit les méta-données sur le disque sans délai et les données utilisateur toutes les 30 secondes.
Quelles sont les informations à mettre à jour sur le disque et quand ?

On a besoin de mettre à jour les informations suivantes sur le disque :
- *contenu de la bitmap qui représente l'espace libre sur le disque*
- *contenu du répertoire courant du fichier (pour rajoter le nouveau fichier)*
- *le nouvel entête fichier pour le nouveau fichier*
- *16 blocs de données pour le contenu du nouveau fichier*
- *un bloc indirect pour pointer vers les 6 derniers blocs du fichier*

Pour assurer la cohérence des méta-données dans le cas d'une panne, Unix écrit immédiatement sur le disque tous les blocs autres que ceux des données du fichier :
- *contenu de la bitmap de l'espace libre sur le disque*
- *le bloc contenant le nouvel entête du fichier*
- *un bloc indirect pointant vers les 6 derniers blocs de données du fichier*
- *le bloc disque du fichier représentant le répertoire courant dans lequel une nouvelle entrée pour le fichier a été rajoutée*

5 – Dans la version Unix BSD 4.2, des changements ont été apportés à la taille des blocs du disque :
a – Pourquoi était-il souhaitable d'augmenter la taille des blocs du disque ?
b – Pourquoi était-il aussi souhaitable de ne pas augmenter la taille des blocs du disque ?
c – Comment ce conflit a été résolu ?

a – La raison principale était d'augmenter le taux de transfert de données. D'autres raisons étaient l'augmentation de l'espace disque utilisable et la diminution de la taille de la bitmap des blocs libres
b –Comme la majorité des fichiers sont de petite taille, de grands blocs disque pouvent faire perdre beaucoup d'espace disque à cause de la fragmentation interne
c – Conflit résolu par 2 moyens :
- introduction de la notion de groupes de cylindres pour ne pas éparpiller les données et réduire ainsi le temps de positionnement de la tête lecture-écriture du disque.
- augmentation de la taille des blocs du disque mais en permettant de diviser chaque bloc en petits fragments si nécessaire.