L2 - *« Programmation système »* *L. Mascarilla,* *E. Zahzah* *V. Owczarek*	**TD n°1 - Processus-Fichiers** **Corrigé**	UNIVERSITÉ La Rochelle

Exercice 1. Fork loop

En reprenant l'exercice 3 du TP1, calculez le nombre de processus fils créés.

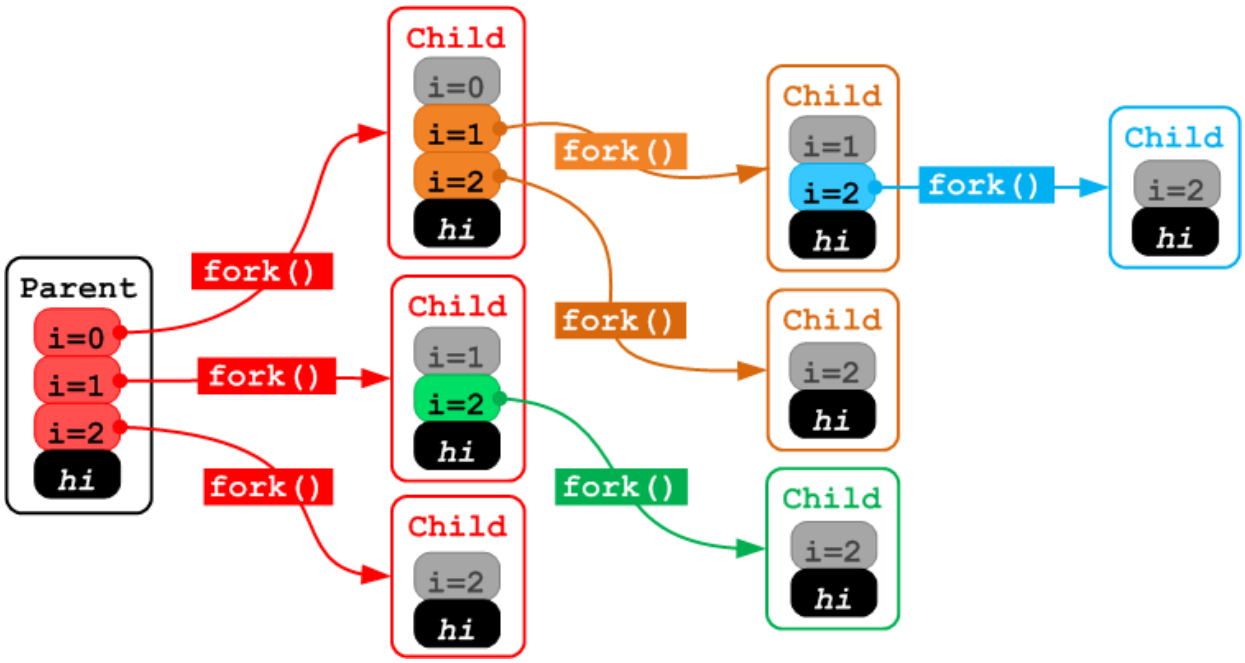
```
#include <stdio.h>

void main()
{
   int i;

   for (i=0;i<3;i++)
   {
      fork() ;

      printf("[%d] [%d] i=%d\n", getppid(), getpid(), i);
   }

   printf("[%d] [%d] hi\n", getppid(), getpid());
}
```



Exercice 2. I-Node

On considère un système disposant d'un système de fichiers « similaire » à celui du ext2 de Linux avec une taille de blocs de données de b=0.5KiO (512 octets) et des pointeurs (numéros de blocs) définis sur 32 bits (4 octets).

On suppose que le i-node de chaque fichier compte 12 pointeurs directs, 1 pointeur indirect simple, 1 pointeur indirect double et 1 pointeur indirect triple.

Combien y a-t-il de pointeurs (de blocs) dans la structure de l'i-node ?

12+1+1+1 = 15 pointeurs

Combien d'adresses peut contenir un bloc de données ?
b/4=128

Quelle est la taille maximale des fichiers adressables par ext2 ?
(12+b/4+(b/4)^2+(b/4)^3)*b # 1Gio

Comment augmenter cette taille ?
Avec b=1024 #16Gio
Attention il existe une limite due à la structure des inodes.

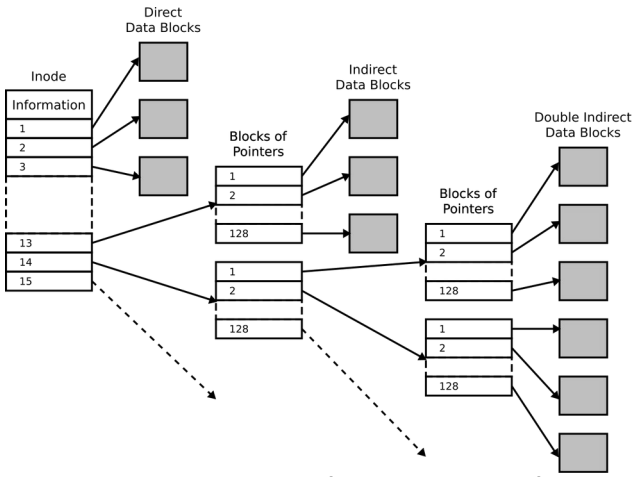
On désire créer un fichier contenant un total de 20.000.000 (vingt millions) de caractères (caractères de fin de ligne et de fin de fichier compris).

- Quelle est la fragmentation interne totale sur le disque résultant de la création de ce fichier ?
- Quelle est la fragmentation externe ?

Nombre de blocs : 20E6/512 # 39062.5 soit 39062 blocs plein + 1 partiellement rempli au total (256 octets)

20E6-39062*512 = 256 octets utilisés dans le dernier bloc
512-256 = 256 octets perdus par fragmentation interne

39062 = 12 (direct) + 128 (simple) + 16384 (double)



Quelle est la proportion de l'espace qui est utilisé pour les données par rapport à l'ensemble des blocs destinés au fichier dans le cas d'un fichier de 71 Kio? Considérez que l'inode occupe un bloc.

Le nombre de blocs de données nécessaires pour stocker le fichier est 71 Kio / 0,5 Kio = 142 blocs. Les pointeurs directs stockent 12 blocs, ce qui laisse 130 blocs à placer dans les autres niveaux. Le nombre de blocs adressables par le pointeur de niveau 1 est de 512 / 4 = 128, il reste donc 130-128 = 2 blocs à placer au niveau 2. Pour adresser 2 blocs depuis le pointeur indirect de niveau 2, il faut un bloc ayant une entrée vers un bloc qui contient 2 pointeurs vers les blocs de données. L'inode occupe 1 bloc, le premier niveau indirect nécessite un bloc et le second niveau indirect nécessite 2 blocs, pour un total de 4. La proportion utilisée pour les données est donc 142 / (142 + 4) = 97%.

Considérez la structure d'inode de la question précédente. Calculez le temps moyen d'accès à chacun des blocs de données pour un fichier de taille maximale et des accès aléatoires dans le cas où il n'existe aucune cache des blocs, et que la lecture d'un bloc nécessite un délai d = 6 ms.

Le nombre maximal de blocs est 2113676 blocs. Le temps d'accès pour les blocs directs nécessite la lecture de l'inode puis le bloc de données (2*d), le premier niveau indirect consiste en l'inode, le bloc indirect puis le bloc de données (3*d), et le second niveau indirect consiste à un accès de plus que le précédent (4*d), le troisième niveau (5*d). Pour la taille maximale et un accès aléatoire, le temps moyen d'accès est donc : ((2*d*12)+ + (3*d*128) + (4*d*128*128)+(5*d*128^3))/2113676, ce qui donne environ 29,9 ms.

Exercice 3. Table des blocs libres

Calculez la taille de la table de bit (bitmap) nécessaire pour stocker les blocs libres/occupés dans la configuration précédente (bloc = 512 octets, adresse sur 32 bits,disque de 8Gio).
Combien de blocs cela représente t-il ?
Quel est le pourcentage du disque qui est occupé par cette table ?
Combien de blocs libres pourrait-on représenter avec ce même nombre de bloc ?
Commentaires ?

2113676 blocs / 8 = 264210 octets soit 516.04 = 517 blocs
517/2113676=0.024460 %

517*512/8 = 66176
66176/2113676=3.13 %
Pas d'espace « gaspillé » systèmatiquement
difficulté à trouver des blocs contigüs de taille fixée
Pas besoin de traverser toute la liste.

Exercice 4. liens

Décrivez les avantages et les inconvénients d'un lien symbolique par rapport à un lien matériel. Dans quels cas chacun d'eux est-il utilisé ?

Un lien symbolique permet de créer des liens entre des fichiers sur deux systèmes de fichiers ayant leur propre liste d'inode tandis qu'un lien dur ne peut franchir la partition du fichier lié. Le lien dur a l'avantage de ne jamais être brisé, contrairement à un lien symbolique qui le deviendra si le fichier lié est supprimé. Les liens symboliques sont généralement utilisés pour les liens entre des noms de librairies sur le système (voir /usr/lib), ce qui permet de déplacer une librairie sur une autre partition. Les liens durs peuvent être utilisés pour les sauvegardes incrémentales, comme dans le système TimeMachine d'Apple.