Programmation système avancée
TP n° 3 : Descripteurs de fichiers et fork

Exercice 1 :

1. Devinez le contenu du fichier `file` après l'exécution de chacun des fragments de code suivants (on suppose que le fichier `file` n' existait pas avant l'exécution du programme). Écrivez un programme pour tester chacune de vos hypothèses. (Il faudra naturellement tester les résultats d'erreur, ce qui n'est pas fait dans les fragments ci-dessous.)

```
fd = open("file", O_RDWR | O_CREAT, 0666);
fd2 = open("file", O_RDWR | O_CREAT, 0666);
write(fd, "a", 1);
write(fd2, "b", 1);
```

```
fd = open("file", O_RDWR | O_CREAT, 0666);
fd2 = dup(fd);
write(fd, "a", 1);
write(fd2, "b", 1);
```

2. Même question pour les fragments de code suivants.

```
pid = fork();
fd = open("file", O_RDWR | O_CREAT, 0666);
write(fd, pid > 0 ? "a" : "b", 1);
```

```
fd = open("file", O_RDWR | O_CREAT, 0666);
pid = fork();
write(fd, pid > 0 ? "a" : "b", 1);
```

3. Même question pour les fragments de code suivants.

```
fork();
fd = open("file", O_RDWR | O_CREAT, 0666);
flock(fd, LOCK_EX);
printf("Lock pris!\n");
while(1)
    pause();
```

```
fd = open("file", O_RDWR | O_CREAT, 0666);
fork();
flock(fd, LOCK_EX);
printf("Lock pris!\n");
while(1)
    pause();
```

Exercice 2 :

Écrivez un progamme qui crée un tube anonyme, puis crée un fils. Le père écrira la chaîne « *coucou* » dans le tube, puis le fermera. Le père lira le contenu du tube jusqu'à épuisement, et affichera les données reçues. Vérifiez à l'aide de « `strace -f` » que le programme a le comportement attendu.

Exercice 3 :

1. Écrivez un programme qui exécute la commande « `w` » et stocke la sortie de celle-ci dans un fichier nommé « `users` ».
2. Écrivez un programme qui affiche le nombre d'utilisateurs connectés en exécutant la commande « `w` » et en comptant le nombre de lignes de la sortie. Votre programme ne devra créer aucun fichier, même temporaire.

Exercice 4 :

Écrivez un programme « `occurrences` » qui prend en paramètre une expression régulière r et n noms de fichiers puis qui affiche le nombre total d'occurrences de r dans les n fichiers en exécutant n occurrences de « grep -c » en parallèle. Le père lira tous les résultats et calculera le nombre total d'occurrences.
(La solution correcte consiste à utiliser un *pipe* pour chaque fils. On peut aussi s'en sortir à l'aide d'un seul *pipe* si l'on fait deux hypothèses raisonnables mais risquées sur le comportement de `grep`. Lesquelles ?)