TP 2: Fork+Exec, fichiers

Exercice 1. Exec et fork

Les fonctions `exec` vue dans le TP2 remplacent le code du processus courant par celui de la commande (`ps` dans l'exercice). Il est donc impossible de récupérer le code retour dans le programme appelant ou d'enchainer des commandes. Pour réaliser cette opération, il suffit d'utiliser `fork` et `exec` ensembles.

Ecrivez un programme *fork_exec.c* qui exécute la commande `ps aux` dans un processus fils grâce à `execlp`. Le père affiche le code de retour du fils.

Exercice 2. Commandes shell synchrones et asynchrones

En utilisant `fork`, `exec` et `wait`, écrire deux programmes équivalents aux commandes shell suivantes :

```
1 uname ; cal ; date -R
```

puis

```
1 uname & cal & date -R
```

Exercice 3. Processus

Créez un programme `pair.c` qui, si le pid du processus est pair crée 2 fils et 1 seul sinon. Chaque fils affichera son `pid`.

Exercice 4. Fichiers

- Ecrivez le programme *cp.c* qui réalise la copie d'un fichier donné en premier paramètre dans un autre fichier donné en second paramètre.

Vous utiliserez les primitives systèmes suivantes : `open`, `read`, `write`, `close`

- Ecrivez le programme *cp_reverse.c* qui réalise la copie d'un fichier donné en premier paramètre dans un autre fichier donné en second paramètre mais en inversant le contenu du fichier. Si le premier fichier contient **abcde**, sa copie contiendra **edcba**.

Vous utiliserez les primitives précédentes ainsi que `lseek`.

Remarque : essayez d'être efficace et de faire la copie par bloc et non octet par octet.