Linux 2 : Appels Systèmes - BINV2181 (tp02- fork/exec/wait)

Les premiers exercices de cette séance ont pour but de vous faire comprendre l'appel système **`fork`**. Respectez scrupuleusement les consignes qui vous sont proposées et expliquez en détails les résultats obtenus, ne passez pas une étape sans vous être assuré qu'elle est bien comprise.

2.A. Exercice sur l'appel système : `fork`

CONSIGNE : Pour cet exercice, ne vous basez pas sur les exemples de code fournis (plus complexes que ce qui est demandé ici) et ne faites pas appel aux fonctions `fork_and_run()` fournies par le module *utils*.

a) Ecrivez un programme qui définit une variable « a » de type `int` et l'initialise à 5 ; crée un processus enfant ; affiche la valeur de retour de l'appel système `fork`. Ensuite, il définit dans l'enfant une nouvelle variable « b » de type `int`, lui affecte la valeur de « a * 2 » puis affiche le contenu de a et de b ; le parent définit une autre variable « b » de type `int`, lui affecte la valeur de « a * 5 » puis affiche le contenu de a et de b.
Pour cet exercice, l'affichage peut se faire avec `printf`.

Qu'affiche ce programme lors de son exécution ?

b) Ecrivez un programme qui affiche la chaîne de caractères "trois .. deux .. un .." avant de créer un processus enfant qui affiche la chaîne de caractères "partez !".
Les écritures doivent se faire grâce à l'appel système `write`.

Qu'affiche ce programme lors de son exécution ?

c) Ajoutez '\n' à la fin des deux chaînes de caractères.

Qu'affiche ce programme modifié ?

d) Refaites les étapes b et c en remplaçant les appels système `write` par l'appel à la fonction `printf` sans '\n' dans les chaines.

Qu'affiche cette première version du programme lors de son exécution ?

Qu'affiche ce programme si les chaînes de caractères se terminent par '\n' ?

e) Y-a-t-il une différence de comportement entre les deux versions ? Si oui, expliquez pourquoi puis modifiez ce qu'il faut pour régler le problème de la première version (sans \n).

f) En conclusion, quelle est la différence qui vient d'être soulignée entre l'appel système `write` et la fonction `printf` ?

2.B. Exercice sur les appels système : `fork` et `wait`

CONSIGNE : Dans cet exercice et le suivant, vous pouvez utiliser les fonctions `fork_and_run()` fournies dans le module *utils*.

a) Ecrivez un programme qui crée un processus enfant qui affiche la string "4 5 6\n" alors que le parent affiche la string "1 2 3\n".

   Testez ce programme et expliquez le résultat obtenu.

b) Modifiez votre programme afin que l'enfant affiche toujours "4 5 6\n" avant que le parent n'affiche "1 2 3\n".

c) Modifiez votre programme afin que l'enfant se termine par un `exit`, puis affiche l'exit code de ce processus enfant. Consultez le man de `waitpid` pour trouver comment récupérer l'exit code du processus enfant.

d) Réécrivez le programme précédent en utilisant la librairie *utils* (présente sur Moovin). Celle-ci devrait vous faciliter la tâche et augmenter la clarté du code !

2.C. Exercice sur les appels système : `fork` et `exec`

Vous allez écrire un programme interactif qui, au final :

- Saisira au clavier le nom d'un fichier script ;
- Créera le script avec les droits `RWX------`. Si le fichier existe, il sera réinitialisé.
- Exécutera un `ls -l` de ce fichier script via un `exec` ;
- Ecrira le shebang dans le script : `#!/bin/bash`
- Saisira au clavier, ligne par ligne, le contenu du script en l'écrivant en parallèle dans le fichier ;
- Exécutera ensuite un `cat` du contenu du script via un `exec` ;
- Exécutera finalement le script en question via un `exec`.

Suivez l'ordre des points **ci-dessous** en testant le caractère fonctionnel de votre code de temps en temps.

a) Effectuez la lecture du nom du fichier script au clavier, à l'aide d'un appel système `read`, sans vérification particulière. Imposez un nom de maximum 20 caractères. Vous pouvez considérer que l'utilisateur ne fera pas de fausses manœuvres.

b) Ouvrez le fichier en question en l'écrasant, ou créez-le si nécessaire.

c) A l'aide d'un `fork`/`exec`, effectuez un `ls -l` sur le nom du script.

d) A l'aide d'appels système `read` et `write`, lisez au clavier, ligne par ligne, le contenu du script en l'écrivant en parallèle dans le fichier créé.

e) A l'aide d'un `fork`/`exec`, effectuez un `cat` du contenu du script.

f) Libérez les ressources, puis exécutez le contenu du script.

g) Les affichages à l'écran peuvent être réalisés grâce à la fonction `printf`.

h) Très important : vous avez à votre disposition la librairie *utils*. utilisez celle-ci à bon escient pour vous faciliter le travail et augmenter la clarté du code !