TP n°3

Fonction *exec*

Pour toute question pendant le TP, commencez par consulter l'aide de `exec` en entrant la commande `man 3 exec`.

Exercice 1 [Rappel d'UNIX : gestion des processus]

Se familiariser avec les différentes options de la commande `ps` en lisant `man ps`.

1. Quel est le processus de pid 1 ?
2. Lancer le programme ci-dessous avec les arguments 10 20. Tapez ps -la dans un autre terminal avant la fin du père, avant la fin du fils. Quels sont les ppid du pere et du fils ? Donnez une explication.
3. Lancer le programme ci-dessous avec les arguments 10 0. Tapez ps -la dans un autre terminal avant la fin du père. Que constatez-vous ?

début src/ex1.c

```
#include <unistd.h> /* necessaire pour les fonctions exec */
#include <sys/types.h> /* necessaire pour la fonction fork */
#include <unistd.h> /* necessaire pour la fonction fork */
#include <stdio.h> /* necessaire pour la fonction perror */

int main(int argc, char * argv[]) {

  pid_t pid;
  int attente_fils,attente_pere;
  if(argc != 3)
    perror("usage: ex1 n m\n");

   attente_pere = atoi(argv[1]);
   attente_fils = atoi(argv[2]);

  switch(pid=fork()) {
  case -1:
    perror("fork error");
    break;
  case 0:
    sleep(attente_fils);
    printf("fils attente finie\n");
    break;
  default:
    sleep(attente_pere);
    printf("pere attente finie\n");
    break;
```

```
  }

}
```

fin src/ex1.c

Exercice 2 [Comportement de `exec`]

1. Faites `exec ps`. Que se passe-t-il ? Donnez une explication.
2. Vérifiez votre explication en étudiant la commande shell `exec sh`. (Quel est le `pid` de ce shell ?)

Exercice 3 [Path et variables d'environnement]

1. Créez dans `progs` un programme `affichez` qui affiche une chaîne de caractères passée en argument.
2. Placez vous dans `tp3` et tapez `affichez salut`. Que se passe-t-il ?
3. Comment faire pour que cette commande exécute le programme `affichez` ?
   Pour vous aidez, demandez vous :
   Que donne la commande shell `which ls` ?
   Que donne la commande shell `echo $PATH` ?

Exercice 4 Ecrire un programme `prog1` qui crée un processus fils qui exécute `affichez` avec l'argument `salut`. On utilisera la fonction `execl`.

Exercice 5 Ecrire un programme `prog2` qui crée un processus fils qui a un environnement à seulement deux variables PATH=/src et USER=unknown. Pour vérification, il affichera ses arguments et ses variables d'environnement. On utilisera cette fois la fonction `execve`.

On s'inspirera du programme ci-dessous qui affiche les valeurs des variables d'environnement de votre machine.

début src/env.c

```
#include <stdio.h> /* necessaire pour la fonction perror */

int main() {

  char ** ptr;
  extern char **environ;

  for(ptr = environ;*ptr != 0;ptr++)
    printf("%s\n",*ptr);

}
```

fin src/env.c

Exercice 6 [Fonction `wait`]

Ecrire un programme qui crée 2 processus l'un faisant la commande `ls -l`, l'autre `ps -l`. Le père devra attendre la fin de ses deux fils et afficher quel a été le premier processus à terminer.