IUT Info, S4, Info2#, système, TP3, enseignant : M. Nakechbandi.
Recouvrement & Duplication de processus (fork + exec)

Rappels
La méthode standard pour qu'un processsus exécute une commande est :
1. le processus crée un processus fils le processus
2. fils effectue un recouvrement pour exécuter la commande
3. a) le processus père attend la fin du processus fils, il peut alors récupérer le code de retour du fils
   b) le processus père n'attend pas la fin du fils, il peut alors générer un 2ème fils pour exécuter une 2ème commande et ainsi de suite. Toutes ces commandes s'exécutent alors en concurrence.

Exercice 1. On considère le programme de exercice 5 TP1 dont l'exécution est donné par le schéma ci-dessous :

```
// tp1 Exercice 5
#include <stdio.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
main (void)
{ pid_t pid ;  int status,i ;
for (i=0 ; i < 4 ; i++) {
switch (fork ()) {
case -1 : perror (" fork ") ;   exit (1) ;
case 0 : /* le fils */
  printf("  Processus %d fils de %d \n ",getpid(),getppid());
   exit (i+1); /* les entiers de 1 à 4 sont les codes de retour
       de exit */
default : /* le pere */
  printf (" pere a cree processus %d\n ", getpid()) ;
  pid=wait (&status) ;
  if (WIFEXITED (status))
   printf (" fils (%d) envoi l'entier : status = %d, le pere
       l'affiche \n ",pid, WEXITSTATUS(status)              ) ;
          }
}
}
```

Résultat de l'execution :

```
pere a cree processus 24771
  Processus 24772 fils de 24771
  fils 24772 envoi l'entier : status = 1, le pere l'affiche
pere a cree processus 24771
  Processus 24773 fils de 24771
  fils 24773 envoi l'entier : status = 2, le pere l'affiche
pere a cree processus 24771
  Processus 24774 fils de 24771
  fils 24774 envoi l'entier : status = 3, le pere l'affiche
pere a cree processus 24771
  Processus 24775 fils de 24771
  fils 24775 envoi l'entier : status = 4, le pere l'affiche
```

On remarque que : La valeur de la variable status est transmis du fils au père via exit(i +1) et via la primitive WEXITSTATUS(status)

Question 1. Récupérer ce programme et l'exécuter (corton --> /tmp--> info4#-->...)
Question 2. Modifier ce programme pour que chaque fils exécute un `ls`. Indication : Insérer `args[0]= "ls";args[1]= NULL; execv("/bin/ls", args);` juste avant `exit(i+1)` et ne pas oublier de déclarer `char * args[3] ;`
Question 3. Concernant la valeur de i+1 transmit du fils au père via `exit(i+1)` et `WEXITSTATUS(status)`. Que constatez-vous ? A-t-on le même résultat que précédemment (question 1) ? Pourquoi ?

Exercice 2. Les 3 programmes ci-dessous permettent de tester les primitives `strtok` et `getenv`

`// teste1_strtok.c`

```
#include <stdio.h>
#include <stdlib.h>  /* pour utiliser getenv */
#include <unistd.h>
#include <string.h>
#include <pwd.h>
int main (void)
{
        char *test;
        char s[]= " Salut tout le mond : Hay everyone ";
        test = strtok (s, ":");
        printf ("%s\n", test);
        test = strtok (NULL, ":");
        printf ("%s\n", test);
        return 0;
}
```

// teste2_strtok.c

```
#include <stdio.h>
#include <unistd.h>
#include <string.h>
#include <pwd.h>
#define Length 100
int main ()
{   char str[] =" ceci; est un simple; string a decouper; on utilise; le
point virgule comme separateur";
  char * pch;
  printf ("\n Splitting string \"%s\" into tokens:\n",str);
  pch = strtok (str,";");
  while (pch != NULL)
  {    printf ("%s\n",pch);     pch = strtok (NULL, ";");
  }
  return 0; }
```

// affiche_path.c

```
#include <stdio.h>
#include <stdlib.h>  /* pour utiliser getenv */
#include <unistd.h>
#include <string.h>
#include <pwd.h>
int main(void)
{
   char *valeur;
   valeur = getenv("PATH");
   if (valeur != NULL)
      printf("Le PATH vaut : %s\n", valeur);
    return 0;
}
```

Qusestion 1. Récupérer ces programmes et les tester
Qusestion 2. S'inspirer du programme `affiche_path.c` pour écrire une commande **`tp3_ex2_2`** qui liste tous les répertoires de la variable *PATH*. **Indication :**

```
//  tp3 ex2_2.c
#include <stdio.h>
#include <stdlib.h>  /* pour utiliser getenv */
#include <unistd.h>
#include <string.h>
#include <pwd.h>
#define Longueur 100
// int main(int argc, char * argv[])
int main(void)
{
   char str[Longueur]  ;
   char * s;
   char * valeur;
   char * args[Longueur] ; /* tableau qui pour la liste des arguments */
   valeur = getenv("PATH");
   int i,j;
   for (i=0 ; i<strlen(valeur) ; i++)   str[i] = valeur[i];
   str[i]=NULL;
   printf ("\n str= \"%s\" \n", str);
      args[0]="ls";
      for(j=1, s=str; (s=strtok(s,":")) != NULL; j++, s=NULL)
         args[j]=s;
         args[j]=NULL;
         execvp(args[0], args);
  return 0;
}
```

Qusestion 3. Modifier le programme précédent pour que un ensemble de processus participe à l'affichage des répertoires de la variable path (un processus par affichage).