TD n°5 : Processus

SOLUTIONS (ne pas distribuer aux étudiants)

Exercice 1 – Création et synchronisation de processus fils

Solution :

1)

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main(void)
{  pid_t pid;
   int i;

   if ((pid = fork()) == -1)
   {  perror("fork"); exit(1);
   }
   if (pid == 0)
   {  /* fils1 */
      for (i = 1; i <= 10; i++)
      {
         printf("%d\n", i);
         sleep(1);
      }
      return 0;
   }
   if ((pid = fork()) == -1)
   {  perror("fork"); exit(1);
   }
   if (pid == 0)
   {  /* fils2 */
      for (i = 11; i <= 20; i++)
      {
         printf("%d\n", i);
         sleep(1);
      }
      return 0;
   }
   return 0;
}
```

2) Pour synchroniser l'affichage, il suffit d'ajouter `wait` avant la création du second fils.

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys:wait.h>
#include <unistd.h>

int main(void)
{  pid_t pid;
```

```
   int i;

   if ((pid = fork()) == -1)
   {  perror("fork"); exit(1);
   }

   if (pid == 0)
   {  /* fils1 */
      for (i = 1; i <= 10; i++)
      {
         printf("%d\n", i);
         sleep(1);
      }
      return 0;
   }

   wait(NULL);

   if ((pid = fork()) == -1)
   {  perror("fork"); exit(1);
   }
   if (pid == 0)
   {  /* fils2 */
      for (i = 11; i <= 20; i++)
      {
         printf("%d\n", i);
         sleep(1);
      }
      return 0;
   }
   return 0;
}
```

Exercice 2 – Simultanéité vs. séquentialité

Solution :

1) Les commandes séparées par & s'exécutent *simultanément*.

```
#include<stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main(void)
{  pid_t pid;

   if ((pid = fork()) == -1)
   {  perror("fork"); exit(1);
   }
   if (pid == 0)
   {  execlp("who", "who", NULL);
      perror("execlp");
      exit(1);
   }
```

```
   if ((pid = fork()) == -1)
   {  perror("fork"); exit(1);
   }
   if (pid == 0)
   {  execlp("ps", "ps", NULL);
      perror("execlp");
      exit(1);
   }

   execlp("ls", "ls", "-l", NULL);
   perror("execlp");
   exit(1);
}
```

2) Les commandes séparées par `;` s'exécutent *successivement*.

```
#include<stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>

int main(void)
{  pid_t pid;

   if ((pid = fork()) == -1)
   {  perror("fork"); exit(1);
   }
   if (pid == 0)
   {  execlp("who", "who", NULL);
      perror("execlp");
      exit(1);
   }

   wait(NULL);

   if ((pid = fork()) == -1)
   {  perror("fork"); exit(1);
   }
   if (pid == 0)
   {  execlp("ps", "ps", NULL);
      perror("execlp");
      exit(1);
   }

   wait(NULL);

   execlp("ls", "ls", "-l", NULL);
   perror("execlp");
   exit(1);
}
```

Exercice 3 – La commande execvp

Solution :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(int argc, char **argv)
{  if (argc < 2)
   {  fprintf(stderr,"Usage: %s commande [arg] [arg] ...\n", argv[0]);
      exit(1);
   }
   execvp(argv[1], argv + 1);
   perror("execvp");
   exit(1);
}
```