# TD Programmation Système
# Série 3 : Synchronisation par tubes
# Correction

## Exercice 1

Après initialisations et création du tube, création du processus fils, les codes du père et du fils sont :

```
/* Père */

wait() ;
close(p[0]) ;
write(p[1],"abcdef",6) ;
close(p[1]) ;
```

```
/* Fils */

close(p[1]) ;
while( read(p[0],&c,1) != 0)
      {
printf("%c", toupper(c)) ;
      }
close(p[0]) ;
```

Question :

1. Ce code présente une situation d'inter-blocage. En effet deux situations se présentent :
   - Soit le Fils commence, dans ce cas il essaie de lire dans le tube (read()), le tube est vide, le read() est bloquant, le processus fils est alors Bloqué. Quand le père reprend la main il exécute le wait(), il se bloque en attente de la mort du Fils (Inter-blocage)
   - Soit le père commence il exécute le wait(), il se bloque en attente de la mort du Fils. Quand ce dernier reprend l'exécution, il essaie de lire dans le tube (read()), le tube est vide, le read() est bloquant, le processus fils est alors Bloqué (Inter-blocage)
2. Une solution sans inter-blocage

```
int main(int argc, char *argv[] ){
char c;
int pid,statut,tube[2];

if(pipe(tube)==-1){ perror("pipe");
                exit(0);
                }
pid=fork();

if(pid==-1){perror("fork");
          exit(0);
          }
if(pid==0){ /* Fils */
            close(tube[1]);
            while (read(tube[0],&c,sizeof(char))!=0)
             {          printf("%c",toupper(c));
             }
            close(tube[0]);
            }

else{      /* Père */
             close(tube[0]);
             write(tube[1],"abcefg", 6);
             close(tube[1]);
    }
 printf("\n");
 return 0;
}
```

# Exercice 2 : Synchronisation Père/Fils

1. Synchronisation par tubes

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

int main(){
int pid,n=0,tube1[2], tube2[2];

if(pipe(tube1)==-1){ perror("pipe");
                     exit(0);
                   }

if(pipe(tube2)==-1){ perror("pipe");
                     exit(0);
                   }

pid=fork();
 if(pid==-1)       { perror("pipe");
                     exit(0);
                   }

while (n<99)
{
if(pid==0){

            close(tube1[0]);
           close(tube2[1]);
           printf("fils :  ");

           do{n=n+2;
            printf(" %d",n);
             }while(n%5!=0);
           printf("\n");
           write(tube1[1],&n,sizeof(int));
           read(tube2[0],&n,sizeof(int));
             }
else{
       close(tube1[1]);
      close(tube2[0]);

      printf("père :  ");

       read(tube1[0],&n,sizeof(int));
      do{n=n+3;
        printf(" %d",n);
        }while(n%5!=0);
      printf("\n");
      write(tube2[1],&n,sizeof(int));

    }

 }
 return 0;
}
```