# TP n°5: Communication et synchronisation (les tubes)

## CORRECTION

### Exercice 1 – La fonction pipe()

***Solution :***

1) Le processus crée un fils qui lit une chaîne de caractères rentrée au clavier par l'utilisateur et qui l'écrit dans un tube. Le père lit toutes les secondes un caractère du tube, le transforme en majuscule et l'affiche sur la sortie standard (l'écran).

On constate que l'écrivain se termine sur <*Control-d*>, mais pas le lecteur. Le lecteur reste en attente bloquante sur `read(fd[0])`. Et pourquoi cela ? Car lui-même n'a pas fait de `close(fd[1])` et donc potentiellement, quelqu'un peut encore écrire sur le tube *fd*. En général il faut penser à fermer les extrémités d'un pipe que l'on n'utilise pas.

2) Remplacez `!fork()` par un appel à `fork()`.

3)

**pipeexec.c**

```
#include <stdio.h>
#include <unistd.h>
#include <ctype.h>
int main(void)
{
char c;
char buf[3];
int fd [2];
pipe(fd);
if (!fork()) {
      while (read(0, &c,1) > 0)
            write(fd[1], & c, 1);
      fprintf(stderr, "ecrivain fin\n");
}else {
      close(fd[1]);
      sprintf(buf , "%d" , fd[0]);
      execl("./lecteur", "lecteur", buf, NULL);
}
return 0;
}
```

**lecteur.c**

```
#include <stdio.h>
#include <unistd.h>
#include <ctype.h>
#include <stdlib.h>
int main (int argc, char *argv[])
{
char c;
int fd0;
```

```
if (argc>1)
{
      fd0=atoi(argv[1]);
} else
{     fd0=0;
}
fprintf(stderr, "=%d=\n" , fd0);
while (read(fd0, &c, 1  > 0 ) {
      c=toupper(c);
      sleep(1);
      write(1, &c, 1);
}
fprintf(stderr, "lecteur fin\n");
return 0;
}
```