Tubes nommées *vs* tubes anomymes

Exercice 1. Décrivez ce qui se passe pendant l'exécution du programme suivant. Comparez-le avec la version « tubes anonymes »donnée dans le TD précédent.

progr1.c

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:  #include <unistd.h>
 4:  #include <assert.h>
 5:  #include <sys/types.h>
 6:  #include <sys/stat.h>
 7:  #include <fcntl.h>
 8:
 9:  int main(void)
10:  {
11:      int readfd, writefd;
12:
13:      if (mkfifo("le_tube", 0777) == -1)
14:      {
15:          perror("mkfifo");
16:          exit(0);
17:      }
18:
19:      if (fork())
20:      {
21:          writefd = open("le_tube", O_WRONLY);
22:          assert(writefd != -1);
23:
24:          close(STDOUT_FILENO);
25:          dup(writefd);
26:          close(writefd);
27:          execlp("ls", "ls", "-l", NULL);
28:          perror("exec ls");      /* Jaimais atteint sauf erreur */
29:      } else
30:      {
31:          readfd = open("le_tube", O_RDONLY);
32:          assert(readfd != -1);
33:          close(STDIN_FILENO);
34:          dup(readfd);
35:          close(readfd);
36:          execlp("wc", "WOdcount", "-l", NULL);
37:          perror("exec wc");      /* jaimais atteint sauf erreur */
38:      }
39:      exit(EXIT_FAILURE);         /* Jaimais atteint sauf erreur */
40:  }
```

Exercice 2. Quel reproche peut-on faire au programme précédent ? proposer une solution.

Exercice 3. Les deux programmes suivants sont-ils équivalents ? Donnez une commande shell permettant de débloquer la version avec un tube nommé.

progr3a.c

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:  #include <unistd.h>
 4:
 5:  int main(void)
 6:  {
 7:      int tube[2];
 8:      char str[10];
 9:
10:      pipe(tube);
11:      write(tube[1], "bonjour", 8);
12:      read(tube[0], str, 8);
13:      printf("%s\n", str);
14:      close(tube[0]);
15:      close(tube[1]);
16:      exit(0);
17:  }
```

progr3b.c

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:  #include <unistd.h>
 4:  #include <assert.h>
 5:  #include <sys/types.h>
 6:  #include <sys/stat.h>
 7:  #include <fcntl.h>
 8:  #include <sys/wait.h>
 9:
10:  #define OK printf("OK\n");
11:  int main(void)
12:  {
13:      int tube[2];
14:      char str[10];
15:
16:      mkfifo("le_tube", 0777);
17:      tube[0] = open("le_tube", O_RDONLY);
18:      tube[1] = open("le_tube", O_WRONLY);
19:      write(tube[1], "bonjour", 8);
20:      read(tube[0], str, 8);
21:      printf("%s\n", str);
22:      close(tube[0]);
23:      close(tube[1]);
24:      exit(0);
25:  }
```

Serveur de mots

Exercice 4 : serveur de mots. Écrire un programme qui crée deux tubes nommés `question` et `answer` ; chaque fois qu'il reçoit un mot sur le tube `question`, il mélange ses lettres et les renvoie sur le tube `answer`. Hypothèse : le serveur ne s'arrête jamais de travailler.

Exercice 5 : arrêt du serveur. Modifier le programme précédent pour que le serveur s'arrête correctement lorsqu'il reçoit le signal `SIGINT`.

Exercice 6 : un client. Écrire un programme qui prend une phrase en argument, envoie tous ses mots au serveur de mots et affiche les mots rendus par le serveur.

Exercice 7 : plusieurs clients. Que se passe-t'il si l'on exécute plusieurs clients se même temps ?