SY40 TP5 / yann derré

PART I

a. Examinez prg1.c puis recompilez-le en mettant l'instruction close(pfd[1]) en commentaire. Comment expliquez-vous le résultat de l'exécution ?

```
(base) yannderre@MacBook-Pro-2 [14:59:59] [~/Downloads/TP5/part1/sources/output]
-> % ./"prg1"
chaine lue dans le tube : salut a toi                                    /0.0s
(base) yannderre@MacBook-Pro-2 [15:00:12] [~/Downloads/TP5/part1/sources/output]
-> % cd "/Users/yannderre/Downloads/TP5/part1/sources/output"
./"prg1"                                                                 /0.0s
(base) yannderre@MacBook-Pro-2 [15:00:16] [~/Downloads/TP5/part1/sources/output]
-> % ./"prg1"
```

En commandant la ligne 34 close(pdf[0]), le programme ne fonctionne plus et n'affiche pas son output. Le tube étant ouvert en écriture à deux endroits, le programme ne fonctionne plus.

b. Est-il possible de remplacer la boucle de lecture caractère par caractère en un seul appel read ?

Actuellement, read ne lit qu'un caractère à la fois dans le buffer. On a un define LGMAX nous permettant de definir la taille totale pour lire l'entièreté du buffer.

```
close(pfd[1]);
read(pfd[0],p,LGMAX) != 0;
```

```
(base) yannderre@MacBook-Pro-2 [15:05:57] [~/Downloads/TP5/part1/sources/output]
-> % ./"prg1"
chaine lue dans le tube : salut a toi                                    /0.3s
(base) yannderre@MacBook-Pro-2 [15:05:58] [~/Downloads/TP5/part1/sources/output]
-> %
```

c. Modifiez le programme prg1.c pour que le processus fils écrive dans le tube depuis un autre exécutable.

```c
#include <stdio.h>
#include <sys/wait.h>
#include <string.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

void erreur(const char *msg)
{
	perror(msg);
	exit(1);
}

#define LGMAX 100

int main()
{
	int f;
	char tampon[LGMAX];
	switch (fork())
	{
	case -1:
		erreur("fork");
	case 0:

		f = open("../joe", O_WRONLY, 0666);
		strcpy(tampon, "salut a toi");
		write(f, tampon, strlen(tampon) + 1);
		break;
	default:
		wait(NULL);
		f = open("../joe", O_RDONLY, 0666);
		read(f, tampon, LGMAX);
		printf("chaine lue dans le tube : %s\n", tampon);
	}
}
```

Programme1 écrit dans un tube accessible depuis un autre exécutable via un pipe nommé joe. On peut le créer via un fichier vide via touch, ou mknod qui est plus adapté.

```
-> % ./"new1"
chaine lue dans le tube : salut a toi
```

d. Examinez prg2.c et commentez-le.

Père:

Le père assigne la fonction traitantAbsenceLEcteur comme handler du signal SIGPIPE.

```
47          signal(SIGPIPE,traitantAbsenceLecteur);
```

Le père écrit dans le tube

```
49          write(pfd[1],tampon,TailleTubeMax+1);
```

Le père attend la fin d'éxecution du fils

```
51          wait(NULL);
```

Fils:

```
41          printf("fils : attente RC clavier");
42          getchar();
```

Le fils attend une touche de l'utilisateur.

```
43          read(pfd[0],tampon,1);
```

Le fils lit le tube

```
44          sleep(1);
```

Le fils attend une seconde.

Fin du fils.

Père:

A la fin de l'exec fils, le père écrit dans le tube, réceptino du SIGPIPE

```
52          write(pfd[1],"A",1);
```

Déclenchement du handler, printf passage par…

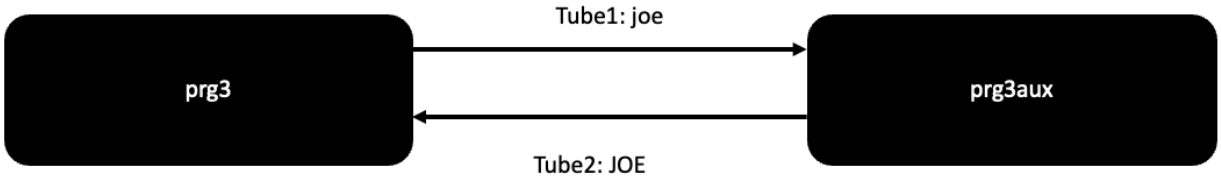
```
void traitantAbsenceLecteur (int num)
  {
  printf("passage par traitantAbsenceLecteur\n");
  exit(1);
  }
```

Fin du programme.

PART II

a. Examinez prg3.c et faites un schéma représentant le mécanisme mis en œuvre.

Deux tubes sont créés permettant à prog3 et prog3aux communiquent dans le sdeux sens. Prog3 exécute prog3aux qui va renvoyer une verison en majuscule du message reçu de tube1 dans tube2.



Le père affiche ensuite le contenu du message reçu et recommence.

b. Ecrivez un programme prg4.c qui reproduit ce que fait le Shell en interprétant la ligne de commande ps | wc -l

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


void erreur(const char* msg)      {perror(msg);exit(1);}

#define LGMSG 20

int main()
  {
    int tubef[2];

    if (pipe(tubef) == -1) erreur("pipe tubef");

    switch (fork())
    {
    case -1:
        erreur("fork");
        break;
    case 0:
        close(tubef[0]);
        dup2(tubef[1], STDOUT_FILENO);
        close(tubef[1]);
        if(execlp("ps", "ps", NULL) == -1){
            erreur("execlp");
        }
        break;
    default:
        wait(NULL);
        close(tubef[1]);
        dup2(tubef[0], STDIN_FILENO);
        close(tubef[0]);
        if(execlp("wc", "wc", "-l", NULL) == -1){
            erreur("execlp");
        }
        break;
    }
  }
```

On utilise dup2 pour rediriger la sortie et l'entrée standard dans le tube.

```
● -> % cd "/Users/yannderre/Downloads/TP5/part2/sources/output"
  (base) yannderre@MacBook-Pro-2 [16:01:39] [~/Downloads/TP5/part2/sources/output]
● -> % ./"prg4"
       3
```

```
  (base) yannderre@MacBook-Pro-2 [16:01:49] [~/Downloads/TP5/part2/sources/output]
● -> % ps | wc -l
       3
```