Univérsité Cheikh Anta Diop

Faculté des Sciences et Techniques
Département Mathématique et Informatique
Année Universitaire 2021-2022
MS2E
Travaux Pratiques de Système d'Exploitation

Atelier 03 : *Programmation des Tubes de Communication UNIX/Linux*

Dr Mandicou BA

Dans ce TP, pour chacun des exercices, il est demandé d'étudier le code source proposé. Après avoir compilé et exécuté chaque programme, analyser et interpréter le résultat obtenu.

Exercice 1 : Communication entre processus père et fils

```c
/**
 * @author Dr Mandicou BA
 * @version 06/02/2017
 */
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/types.h>

void fils(int tube[2]) {
  int i, tmp;

  if(close(tube[1]) == -1) {
    perror("Fils : erreur lors de la fermeture du tube en ecriture ");
    exit(EXIT_FAILURE);
  }

  for(i = 0; i < 5; i++) {
    if(read(tube[0], &tmp, sizeof(int)) == -1) {
      perror("Fils : erreur lors de la lecture d'un entier depuis le tube ");
      exit(EXIT_FAILURE);
    }
    printf("Fils : entier lu : %d\n", tmp);
  }

  if(close(tube[0]) == -1) {
    perror("Fils : erreur lors de la fermeture du tube en lecture ");
    exit(EXIT_FAILURE);
  }

  printf("Fils : termine.\n");

  exit(EXIT_SUCCESS);
}

int main() {
  pid_t pid;
  int tube[2], i;
```

```
  if(pipe(tube) == -1) {
    perror("Pere : erreur lors de la creation du tube ");
    exit(EXIT_FAILURE);
  }

  if((pid = fork()) == -1) {
    perror("Pere : erreur lors de la creation du fils ");
    exit(EXIT_FAILURE);
  }
  if(pid == 0)
    fils(tube);

  if(close(tube[0]) == -1) {
    perror("Pere : erreur lors de la fermeture du tube en lecture ");
    exit(EXIT_FAILURE);
  }

  for(i = 0; i < 5; i++) {
    if(write(tube[1], &i, sizeof(int)) == -1) {
      perror("Pere : erreur lors de l'ecriture d'un entier dans le tube ");
      exit(EXIT_FAILURE);
    }
    printf("Pere : entier envoye : %d\n", i);
    sleep(1);
  }

if(close(tube[1]) == -1) {
    perror("Pere : erreur lors de la fermeture du tube en ecriture ");
    exit(EXIT_FAILURE);
  }

  if(waitpid(pid, NULL, 0) == -1) {
    perror("Pere : erreur lors de l'attente de la fin du fils ");
    exit(EXIT_FAILURE);
  }
  printf("Pere : fils termine.\n");

  return EXIT_SUCCESS;
}
```

Exercice 2 : Écriture dans un tube sans lecteur

```
/**
 * @author Dr Mandicou BA
 * @version 06/02/2017
 */
#include <stdlib.h>
#include <signal.h>
#include <sys/stat.h>
#include <errno.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#define _POSIX_SOURCE 1

void sig_handler (int sig) {

    if (sig == SIGPIPE)
```

```
        printf ("ecriture dans un tube sans lecteurs \n");

}

int main (int argc, char ** argv) {

    int tubeDesc[2];
    signal(SIGPIPE,sig_handler);
    if (pipe (tubeDesc) == - 1){
        perror ("Erreur creation pipe");
        exit (1);
    }

    close(tubeDesc[0]);

    if(write(tubeDesc[1],"x", 1) == -1){

            perror ("Erreur de write");
    }


    return EXIT_SUCCESS;



}
```

Exercice 3 : Tube nommé - Modèle Écrivains Lecteur

1. **Partie Écrivains**

```
/**
 * @author Dr Mandicou BA
 * @version 06/02/2017
 */
#include <stdlib.h>
#include <signal.h>
#include <sys/stat.h>
#include <sys/stat.h>
#include <errno.h>
#include <unistd.h>
#include <stdio.h>
#include <fcntl.h>
#define S_BUF 100

int n;

char buffer[S_BUF];

int main (int argc, char ** argv) {
    int fd_write;

    if (mkfifo(argv[1], S_IRUSR|S_IWUSR) == -1) {
        perror("mkfifo");
        exit (1);
    }

    if ((fd_write = open (argv[1], O_WRONLY)) == -1){
        perror ("open");
```

```
        exit (1);
    }

    if ((n= write(fd_write,"Bonjour", 7)) == -1){

        perror ("write");
        exit (1);
    }

    close (fd_write);

    return EXIT_SUCCESS;
}
```

2. **Partie Lecteur**

```
/**
 * @author Dr Mandicou BA
 * @version 06/02/2017
 */
#include <stdlib.h>
#include <sys/stat.h>
#include <errno.h>
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>
#include <fcntl.h>
#define _POSIX_SOURCE 1

#define S_BUF 100

int n;

char buffer[S_BUF];

int main (int argc, char ** argv) {

    int fd_read;

    if ((fd_read = open (argv[1],O_RDONLY)) == -1){
        perror ("Erreur de open");
        exit (1);
    }

    if(( n= read (fd_read, buffer, S_BUF)) == -1){
        perror ("Erreur de read");
        exit (1);
    }
    else {
        buffer[n] = '\0';
        printf ("%s\n",buffer);
    }

    close (fd_read);
    return EXIT_SUCCESS;
}
```