```c
/* les tubes sous Linux */
/* Exercice 1, version avec E/S  Linux */

# include <sys/types.h>
# include <unistd.h>
# include <stdio.h>
# include <string.h>

void traitFils( int tube[2] );

int main()
{
        int unTube[2];
        int  err;
        pid_t  pidProc ;
        char tampon[512];
        int statut;

        /* processus pere */
        /* creation du tube */
        err = pipe( unTube );
        if (  err == -1 )
        { perror("echec pipe");exit(1); }

        /* creation du fils */
        pidProc = fork() ;
        switch  (pidProc )
            {
            case -1 : perror("fork"); exit(1);   /* erreur */

            case 0 :  traitFils( unTube ); exit(2) ;  /* appel du traitement du fils1 */
            }

        /* suite du processus pere */
        /* fermeture du tube en lecture - absolument necessaire !!!!*/
          close( unTube[0] );

        /* lecture d'une suite de chaines au clavier
           et ecriture dans le tube */

        printf("\nPere->Entrer une chaine ou stop pour arreter:\n");
        fgets( tampon, sizeof(tampon), stdin );
        while (strcmp(tampon,"stop\n") != 0 )
        {
              write( unTube[1], tampon, strlen(tampon)+1 );
              sleep(2);
              printf("Pere->Entrer une chaine ou stop pour arreter:\n");
              fgets( tampon, sizeof(tampon), stdin );
        }
        write( unTube[1], tampon, strlen(tampon)+1 );

        /* attente de la terminaison du fils - absolument necessaire !!!!*/
        pidProc=wait( &statut);
}
/* fin du processus pere */


/* traitement du fils */
void traitFils( int tube[2] )
{
        int nbLu;
        char chaine[512];
        FILE * fic;

        /* fermeture du tube en ecriture - absolument necessaire !!!!*/
          close( tube[1] );

        /* ouverture du fichier */
        fic= fopen( "data.txt","w");
        if ( fic == NULL )
           { perror("echec fopen");exit(3); }
```

```
        /* lecture d'une suite de chaines dans le tube
           et ecriture dans un fichier */

        nbLu=read( tube[0], chaine, sizeof(chaine) );
        if ( nbLu == -1 )
           { perror("echec read"); exit(4); }

        while (strcmp(chaine,"stop\n") != 0 )
             {
              printf("Fils->chaine recue: %s", chaine);
              fputs( chaine, fic );

                 nbLu=read( tube[0], chaine, sizeof(chaine) );
              if ( nbLu == -1 )
              { perror("echec read"); exit(4); }
             }

        /* terminaison du fils */
           exit(0);
}

/* les tubes sous Linux */
/* Exercice 1, version avec E/S en C  */

# include <sys/types.h>
# include <unistd.h>
# include <stdio.h>
# include <string.h>

void traitFils( int tube[2]);

int main()
{
        int unTube[2];
        int  err;
        pid_t  pidProc ;
        char tampon[512];
        int statut;
        FILE * tubeQueue;

        /* processus pere */
        /* creation du tube */
        err = pipe( unTube );
        if (  err == -1 )
        { perror("echec pipe");exit(1); }

        /* creation du fils */
        pidProc = fork() ;
        switch  (pidProc )
            {
              case -1 : perror("fork"); exit(1);   /* erreur */
              case 0 :  traitFils( unTube ); exit(2) ;  /* appel du traitement du fils1 */
            }

        /* suite du processus pere */
        /* fermeture du tube en lecture – absolument necessaire !!!!*/
          close( unTube[0] );
        /* creation du descripteur FILE *  */
          tubeQueue=fdopen(unTube[1],"w");

        /* lecture d'une suite de chaines au clavier
           et ecriture dans le tube */

        printf("\nPere->Entrer une chaine ou stop pour arreter:\n");
        fgets( tampon, sizeof(tampon), stdin );
        while (strcmp(tampon,"stop\n") != 0)
            {
              fputs(tampon,tubeQueue);
              fflush(tubeQueue);  /* absolument necessaire !!!!  */
              sleep(2);
              printf("Pere->Entrer une chaine ou stop pour arreter:\n");
```

```
              fgets( tampon, sizeof(tampon), stdin );
            }
        fputs(tampon,tubeQueue);
        fflush(tubeQueue);     /* absolument necessaire !!!!  */

        /* attente de la terminaison du fils – absolument necessaire !!!!*/
        pidProc=wait( &statut);

}  /* fin du processus pere */

/* traitement du fils */
void traitFils( int tube[2] )
{
        char chaine[512];
        FILE * tubeTete, fic;

        /* fermeture du pipe en ecriture – absolument necessaire !!!!*/
          close( tube[1] );

        /* creation du descripteur FILE*  */
          tubeTete=fdopen(tube[0],"r");

        /* ouverture du fichier */
        fic=fopen("databis.txt","w");
        if (  fic == NULL )
           { perror("echec fopen");exit(3); }

        /* lecture d'une chaine dans le pipe    et affichage a l'ecran */
        fgets( chaine, sizeof(chaine), tubeTete );
        while (strcmp(chaine,"stop\n") != 0)
            {
               printf("\nFils -> chaine lue: %s\n", chaine);
               fputs(chaine , fic);
               fgets( chaine, sizeof(chaine), tubeTete );
             }

        exit(0);
} /* terminaison du fils */


/* Exercice 1, version avec E/S  Linux */
/* et transfert d'un nombre fixe de chaines */

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

void traitFils( int tube[2] );

int main()
{
        int unTube[2];
        int  err;
        pid_t  pidProc ;
        char tampon[512];
        int statut, i;

        /* processus pere */
        /* creation du tube */
        err = pipe( unTube );
        if (  err == -1 )
        { perror("echec pipe");exit(1); }

        /* creation du fils */
        pidProc = fork() ;
        switch  (pidProc )
            {
              case -1 : perror("fork"); exit(1);   /* erreur */
              case 0 :  traitFils( unTube ); exit(2) ;  /* appel du traitement du fils1 */
            }
```

```
        /* suite du processus pere */
        /* fermeture du tube en lecture  - absolument necessaire !!!!*/
          close( unTube[0] );

        /* lecture de 3 chaines au clavier et ecriture dans le tube */
            for (i=0; i<3; i++)
            {
              printf("\nPere->Entrer une chaine :\n");
              fgets( tampon, sizeof(tampon), stdin );

              write( unTube[1], tampon, strlen(tampon)+1 );
              sleep(2);
            }
         close( unTube[1] );     /*  - absolument necessaire !!!!  */

        /* attente de la terminaison du fils - absolument necessaire !!!! */

        pidProc=wait( &statut);
}  /* fin du processus pere */


/* traitement du fils */
void traitFils( int tube[2] )
{
        int nbLu;
        char chaine[512];
        FILE * fic;

        /* fermeture du tube en ecriture - absolument necessaire !!!! */
        close( tube[1] );

        /* ouverture du fichier */
        fic=fopen( "data.txt","w");
        if ( fic == NULL )
                { perror("echec fopen");exit(3); }

        /* lecture d'une suite de chaines dans le tube et ecriture dans un fichier */

        nbLu=read( tube[0], chaine, sizeof(chaine) );
        if ( nbLu == -1 )
                { perror("echec read"); exit(4); }

        while (nbLu != 0 )   /* des que le pere ferme le tube en ecriture, il n'y a plus aucun
processus */
                                        /* succeptible d'ecrire  dans le tube, donc des que le tube
est vide  */
                                        /* la fonction read retourne  0  dans le  fils bloqué  en
lecture */
           {
              printf("Fils->chaine recue: %s", chaine);
              fputs( chaine, fic );

              nbLu=read( tube[0], chaine, sizeof(chaine) );
              if ( nbLu == -1 )
                { perror("echec read"); exit(4); }
           }

        close( tube[0] ); /*facultatif*/
        fclose( fic );    /*facultatif*/
        exit(0);
} /* terminaison du fils */


/* les tubes sous Linux */
/* Exercice 2 – fichier signal1-fifo.c  */

#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <signal.h>
#include <stdio.h>
```

```c
#include <fcntl.h>
#include <errno.h>

typedef enum {FAUX=0,VRAI=1} booleen;
extern int errno;

void traiterSIGUSR1( int sig )
{
        switch ( sig )
            {
            case SIGUSR1 :
                printf("\nPID= %d\n", getpid() );
                printf("n° du signal reçu = %d\n", sig );
                break;
            default :
                printf("\n Erreur système !!!!\n");
            }
        exit(2);
}

int main ( )
{
        int err;
        int fifo;
        pid_t pid;
        void (*errSig)() ;

        FILE * fifoC;     /* declaration a rajouter si on utilise les E/S en C */

        err=mkfifo("file.FIFO",0600);    /* creation du FIFO  */
            if (err == -1)
                if ( errno != EEXIST) { perror("echec mkfifo"); exit(1); }

        fifo=open("file.FIFO", O_WRONLY);
        if (fifo == -1) { perror("echec open"); exit(2); }

        /* solution avec les E/S unix */
        pid=getpid();
        write(fifo, &pid, sizeof(pid_t));

        /* autre solution avec les E/S en C */
        /*  fifoC=fdopen(fifo, "w");
             if (fifoC == NULL) { perror("echec fdopen"); exit(3); }
             fprintf(fifoC, "%d\n", pid); fflush(fifoC);  */

        errSig=signal( SIGUSR1, traiterSIGUSR1 );
        if (errSig == SIG_ERR) { perror("echec signal"); exit(4); }

        while (VRAI)
          { sleep(5); }
}


/* les tubes sous Linux */
/* Exercice 2 – fichier signal2-fifo.c */

#include <sys/types.h>
#include <unistd.h>
#include <signal.h>
#include <stdio.h>
#include <fcntl.h>

int main()
{
        int err;
        int fifo;
        pid_t pid;

        FILE * fifoC;      /* declarations a rajouter */
        char tampon[256];  /* si on utilise les E/S en C */

        fifo=open("file.FIFO", O_RDONLY);
```

```
        if (fifo == -1) { perror("echec open"); exit(1); }

        /* solution avec les E/S Unix  */
        read(fifo, &pid, sizeof(pid) );

        /* autre solution avec les d'E/S en C */
        /*   fifoC=fdopen(f, "r");
             if (fifoC == NULL) { perror("echec fdopen"); exit(2); }

             fgets(tampon, sizeof(tampon), fifoC);
             pid=atoi(tampon);    */

        printf("envoi du signal SIGUSR1 au processus= %d\n", pid);

        err=kill( pid, SIGUSR1 );
        if (err == -1) { perror("echec kill"); exit(3); }
}
```