```c
// OPERATING SYSTEMS
// PIPES- Examples from lectures
// Father process reads 2 integers from the keyboard
// and sends them to its son, through a pipe.
// Son reads the integers from the pipe,
// computes their sum and displays the result.
// JAS
// pipe00.c

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

#define READ 0
#define WRITE 1

int main(void)
{
  int fd[2];
  pid_t pid;

  pipe(fd);

  pid = fork();

  if (pid >0) //pai
  {
    int a[2];
    printf("PARENT:\n");
    printf("x y ? "); scanf("%d %d",&a[0],&a[1]);
    close(fd[READ]);
    write(fd[WRITE],a,2*sizeof(int));
    close(fd[WRITE]);
  }
  else //filho
  {
    int b[2];
    //printf("SON:\n");
    close(fd[WRITE]);
    read(fd[READ],b,2*sizeof(int));
    printf("SON:\n"); //WHY HERE AND NOT ABOVE ...?!
    printf("x + y = %d\n", b[0]+b[1]);
    close(fd[READ]);
  }
  return 0;
}
```

```c
// SISTEMAS OPERATIVOS
// PIPES- Exemplos das aulas teóricas
// Programa que mostra um ficheiro, página a página,
// usando o paginador do UNIX
// JAS
// pipe01.c

#include <stdio.h>
#include <stdlib.h>

#define   MAXLINE   1000
#define   PAGER "/bin/more"

int main(int argc, char *argv[])
{
      char   line[MAXLINE];
      FILE   *fpin, *fpout;

      if (argc != 2) { printf("usage: %s filename\n",argv[0]); exit(1); }
      if ((fpin = fopen(argv[1], "r")) == NULL)
      {
            fprintf(stderr,"can't open %s", argv[1]); exit(1);
      }
      if ((fpout = popen(PAGER, "w")) == NULL)
      {
            fprintf(stderr,"popen error"); exit(1);
      }
      /* copy filename contents to pager - file=argv[1] */
      while (fgets(line, MAXLINE, fpin) != NULL)
      {
            if (fputs(line, fpout) == EOF)
            {
                  printf("fputs error to pipe"); exit(1);
            }
      }
      if (ferror(fpin))
      {
            fprintf(stderr,"fgets error"); exit(1);
      }
      if (pclose(fpout) == -1)
      {
            fprintf(stderr,"pclose error");
            exit(1); }
      exit(0);
}
```

```c
// SISTEMAS OPERATIVOS
// PIPES- Exemplos das aulas teoricas
// Programa que mostra um ficheiro, pagina a pagina,
// usando o paginador do UNIX/LINUX
//
// VERSAO SIMPLIFICADA DE pipe01.c (SEM TESTES DE ERRO)
// JAS
// pipe01_s.c
// EXECUCAO: ./pipe01_s pipe01_s.c

#include <stdio.h>
#include <stdlib.h>
#define MAXLINE 1000
#define PAGER "/bin/more"

int main(int argc, char *argv[])
{
  char line[MAXLINE];
  FILE *fpin, *fpout;

  if (argc != 2) { printf("usage: %s filename\n",argv[0]); exit(1); }

  fpin = fopen(argv[1], "r");
  fpout = popen(PAGER, "w");

  while (fgets(line, MAXLINE, fpin) != NULL)
    fputs(line, fpout) == EOF;

  pclose(fpout);

  exit(0);
}
```

```c
//=====================================================================

// SISTEMAS OPERATIVOS
// PIPES- Exemplos das aulas teóricas
// JAS
// Programa que usa um filtro que converte maiúsculas em minúsculas
// pipe02a.c (a executar em conjunto com pipe02b.c = filtro)

#include <stdio.h>
#include <stdlib.h>

#define MAXLINE 1000

int main(void)
{
      char line[MAXLINE];
      FILE *fpin;

      if ((fpin=popen("./pipe02b","r")) == NULL)
      {printf("popen error"); exit(1);}
      for ( ; ; )
      { fputs("prompt > ",stdout); fflush(stdout);
      if (fgets(line,MAXLINE,fpin) == NULL) break;
      if (fputs(line,stdout) == EOF)
      { fprintf(stderr,"fputs error"); exit(1);}
      }
      if (pclose(fpin)==-1)
      {fprintf(stderr,"pclose error"); exit(1);}
      putchar('\n');
      exit(0);
}

//=====================================================================

// SISTEMAS OPERATIVOS
// PIPES - Exemplos das aulas teóricas
// JAS
// Filtro que converte maiúsculas em minúsculas
// pipe02b.c (a executar em conjunto com pipe02a.c)

#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>

int main(void)
{
      int c;

      while ((c=getchar()) != EOF)
      {
            if (isupper(c)) c=tolower(c);
            if (putchar(c)==EOF)
            {
                  printf("output error"); exit(1);
            }
            if (c=='\n') fflush(stdout);
      }
      exit(0);
}
```

```c
//========================================================================

// SISTEMAS OPERATIVOS
// PIPES - Exemplos das aulas teóricas
// JAS (adaptado de Stevens)
// Programa que usa um coprocesso para fazer contas de somar (!)
// pipe03a.c (a executar em conjunto com pipe03b.c = coprocesso)

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

#define MAXLINE 1000
#define READ    0
#define WRITE   1

void sig_pipe(int signo);
void err_sys(char *msg);
void err_msg(char *msg);

int main(void)
{
      int     n, fd1[2], fd2[2];
      pid_t   pid;
      char    line[MAXLINE];

      if (signal(SIGPIPE, sig_pipe)==SIG_ERR)
            err_sys("signal error");
      if (pipe(fd1)<0 || pipe(fd2)<0)
            err_sys("pipe error");
      if ((pid=fork())<0) err_sys("fork error");
      else
            if (pid>0) /* PARENT */
            {
                  close(fd1[READ]); close(fd2[WRITE]);
                  printf("Input 2 numbers (END = CTRL-D): ");
                  while (fgets(line, MAXLINE,stdin) != NULL)
                  {
                        n=strlen(line);
                        if (write(fd1[WRITE],line,n) != n)
                              err_sys("write error to pipe");
                        if ((n=read(fd2[READ],line,MAXLINE)) < 0)
                              err_sys("read error from pipe");
                        if (n==0) {err_msg("child closed pipe"); break;}
                        printf("sum = ");
                        line[n]=0;
                        if (fputs(line,stdout)==EOF) err_sys("fputs error");
                        printf("Input 2 numbers  (END 0 CTRL-D): ");

                  }
                  if (ferror(stdin)) err_sys("fgets error on stdin");
                  exit(0);
            }
```

```
            else        /* CHILD */
            {
                  close(fd1[WRITE]); close(fd2[READ]);
                  if (fd1[READ] != STDIN_FILENO)
                  {
                        if (dup2(fd1[READ],STDIN_FILENO) != STDIN_FILENO)
                              err_sys("dup2 error to stdin");
                        close(fd1[READ]);
                  }
                  if (fd2[WRITE] != STDOUT_FILENO)
                  {
                        if (dup2(fd2[WRITE],STDOUT_FILENO) != STDOUT_FILENO)
                              err_sys("dup2 error to stdout");
                        close(fd2[WRITE]);
                  }
                  if (execl("./pipe03b","pipe03b",NULL) < 0)
                        err_sys("execl error");
            }
            return 0;
}

void sig_pipe(int signo)
{
      printf("SIGPIPE caught\n");
      exit(1);
}

void err_sys(char *msg)
{
      fprintf(stderr,"%s\n",msg);
      exit(1);
}

void err_msg(char *msg)
{
      printf("%s\n",msg); return;
}
```

```c
// SISTEMAS OPERATIVOS
// PIPES - Exemplos das aulas teóricas
// Programa que usa um coprocesso para fazer contas de somar (!)
// pipe03a_s.c (versão simplificada, sem testes de erro, de pipe03a.c)
// (a executar em conjunto com pipe03b.c)
#include <stdio.h>
#include <signal.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>

#define MAXLINE 1000
#define READ    0
#define WRITE   1

void sig_pipe(int signo);
void err_sys(char *msg);
void err_msg(char *msg);

int main(void)
{
      int     n, fd1[2], fd2[2];
      pid_t   pid;
      char    line[MAXLINE];

      signal(SIGPIPE, sig_pipe);
      pipe(fd1);
      pipe(fd2);
      pid=fork();
      if (pid>0) // PARENT
      {
            close(fd1[READ]); close(fd2[WRITE]);
            printf("Input 2 numbers (END = CTRL-D): ");
            while (fgets(line, MAXLINE,stdin) != NULL)
            {
                  n=strlen(line);
                  write(fd1[WRITE],line,n); // null ending char is not send !
                  n=read(fd2[READ],line,MAXLINE); //waits for answer (= sum)
                  if (n==0)
                  {
                        err_msg("child closed pipe"); break;
                  }
                  line[n]=0; // null ending char is not received, so "add" it
                  printf("sum = %s",line);
                  printf("Input 2 numbers  (END 0 CTRL-D): ");
            }
            exit(0);
      }
      else       // CHILD
      {
            close(fd1[WRITE]); close(fd2[READ]);
            dup2(fd1[READ],STDIN_FILENO);    // redirect I/O of the coprocess
            dup2(fd2[WRITE],STDOUT_FILENO);  // to the pipes
            if (execl("./pipe03b","pipe03b",NULL) < 0) // execute the coprocess
                  err_sys("execl error");
      }
      return 0;
}
```

```c
void sig_pipe(int signo)
{
      printf("SIGPIPE caught\n");
      exit(1);
}

void err_sys(char *msg)
{
      fprintf(stderr,"%s\n",msg);
      exit(1);
}

void err_msg(char *msg)
{
      printf("%s\n",msg);
      return;
}
```

```c
//========================================================================

// SISTEMAS OPERATIVOS
// PIPES - Exemplos das aulas teóricas
// JAS (adaptado de Stevens)
// Coprocesso para fazer contas de somar (!)
// pipe03b.c (a executar em conjunto com pipe03a.c)

#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

#define MAXLINE 100

int main(void)
{
      int    n, int1, int2;
      char   line[MAXLINE];

      while ( (n = read(STDIN_FILENO, line, MAXLINE)) > 0)
      {
            line[n] = 0;  // null terminate
            if (sscanf(line, "%d%d", &int1, &int2) == 2)
            {
                  sprintf(line, "%d\n", int1 + int2);
                  n = strlen(line);
                  if (write(STDOUT_FILENO, line, n) != n)
                  {
                        fprintf(stderr,"write error"); exit(1);
                  }
            }
            else
            {
                  if (write(STDOUT_FILENO, "invalid args\n", 13) != 13)
                  {
                        fprintf(stderr,"write error"); exit(1);
                  }
            }
      }
      exit(0);
}
```

```c
//========================================================================
// SISTEMAS OPERATIVOS
// FIFOS - Exemplos das aulas teóricas
//
// PROGRAMA reader
// fifo01a.c
// deve correr conjuntamente com fifo01b.c
// Experimentar:
// 1) ./fifo01a & ./fifo01b & ./fifo01b & ./fifo01b &
// 2) correr fifo01a numa janela de comando e fifo01b noutra
//    e interpretar resultado;
// lancar em execucao primeiro fifo01a e depois fifo01b e depois o inverso

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

int readline(int fd, char *str);

int main(void)
{
      int   fd;
      char  str[100];

      mkfifo("myfifo",0660);
      fd=open("myfifo",O_RDONLY);

      putchar('\n');
      while(readline(fd,str)) printf("%s",str);
      close(fd);
      return 0;
}


int readline(int fd, char *str)
{
      int n;

      do
      {
            n = read(fd,str,1);
      }
      while (n>0 && *str++ != '\0');
      return (n>0);
}
```

```c
//========================================================================
// SISTEMAS OPERATIVOS
// FIFOS - Exemplos das aulas teóricas
//
// PROGRAMA writer
// fifo01b.c
// deve correr conjuntamente com fifo01a.c
// Experimentar:
// 1) ./fifo01a & ./fifo01b & ./fifo01b & ./fifo01b &
// 2) correr fifo01a numa janela de comando e fifo01b noutra
//    e interpretar resultado;
// lancar em execucao primeiro fifo01a e depois fifo01b e depois o inverso


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

int main(void)
{
      int   fd, messagelen, i;
      char  message[100];

      do
      {
            fd=open("myfifo",O_WRONLY);
            if (fd==-1) sleep(1);
      }
      while (fd==-1);

      for (i=1; i<=3; i++)
      {
            sprintf(message,"Hello no. %d from process no. %d\n", i, getpid());
            messagelen=strlen(message)+1;
            write(fd,message,messagelen);
            sleep(3);
      }
      close(fd);
      return 0;
}
```

```c
/* SISTEMAS OPERATIVOS
   Arquitectura cliente-servidor

   Programa servidor - srv_01.c

   O cliente envia o nome do utilizador ao servidor
   e este escreve no ecrã "Username has arrived".

   Servidor faz leitura do FIFO com espera activa.
*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <signal.h>
#include <errno.h>
#include <string.h>

#define MAX_MSG_LEN 20

int main(void)
{
 int   fd,n;
 char  str[MAX_MSG_LEN];

 if (mkfifo("/tmp/requests",0660)<0)
  if (errno==EEXIST) printf("FIFO '/tmp/requests' already exists\n");
  else printf("Can't create FIFO\n");
 else printf("FIFO '/tmp/requests' sucessfully created\n");

 if ((fd=open("/tmp/requests",O_RDONLY)) !=-1)
  printf("FIFO '/tmp/requests' openned in READONLY mode\n");

 do
  {
    n=read(fd,str,MAX_MSG_LEN);               // QUAL É O PROBLEMA DESTA SOLUÇÃO ?
    if (n>0) printf("%s has arrived\n",str);  // O QUE ACONTECE Q.DO NÃO HOUVER CLIENTES ?
    sleep(1);                                 // COMO RESOLVÊ-LO ?
  } while (strcmp(str,"SHUTDOWN")!=0);

  close(fd);
  if (unlink("/tmp/requests")<0)
   printf("Error when destroying FIFO '/tmp/requests'\n");
  else
   printf("FIFO '/tmp/requests' has been destroyed\n");
  exit(0);
}
```

```c
/* SISTEMAS OPERATIVOS
   Arquitectura cliente-servidor

   Programa cliente  - cli_01.c = cli_02.c

   O cliente envia o nome do utilizador ao servidor
   e este escreve no ecrã "Username has arrived"
*/

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

int main(int argc, char *argv[])
{
 int   fd;

 if (argc!=2) {
  printf("Usage: cli_01 <username>\n");
  exit(1);
 }

 fd=open("/tmp/requests",O_WRONLY);
 if (fd == -1) {
  printf("Oops !!! Server is closed !!!\n");
  exit(1);
 }

 printf("FIFO 'requests' openned in WRITEONLY mode\n");

 write(fd,argv[1],strlen(argv[1])+1);
 close(fd);
 return 0;
}
```

```c
/* SISTEMAS OPERATIVOS
   Arquitectura cliente-servidor

   Programa servidor - srv_02.c

   O cliente envia o nome do utilizador ao servidor
   e este escreve no ecran "Username has arrived".

   Servidor faz leitura do FIFO sem espera activa.
*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <signal.h>
#include <errno.h>
#include <string.h>

#define MAX_MSG_LEN 20

int main(void)
{
 int   fd, n, fd_dummy;
 char  str[MAX_MSG_LEN];

 if (mkfifo("/tmp/requests",0660)<0)
  if (errno==EEXIST) printf("FIFO '/tmp/requests' already exists\n");
  else printf("Can't create FIFO\n");
 else printf("FIFO '/tmp/requests' sucessfully created\n");

 if ((fd=open("/tmp/requests",O_RDONLY)) !=-1)
  printf("FIFO '/tmp/requests' openned in READONLY mode\n");

 // UMA SOLUÇÃO P/PROBLEMA DE srv_01.c (busy waiting)
 // EXISTE OUTRA SOLUÇÃO ?
 if ((fd_dummy=open("/tmp/requests",O_WRONLY)) !=-1)
  printf("FIFO '/tmp/requests' openned in WRITEONLY mode\n");

 do
  {
    n=read(fd,str,MAX_MSG_LEN);
    if (n>0) printf("%s has arrived\n",str);
  } while (strcmp(str,"SHUTDOWN")!=0);

  close(fd);
  close(fd_dummy);
  if (unlink("/tmp/requests")<0)
   printf("Error when destroying FIFO '/tmp/requests'\n");
  else
   printf("FIFO '/tmp/requests' has been destroyed\n");
  exit(0);
}
```

```c
/* SISTEMAS OPERATIVOS
   Arquitectura cliente-servidor

   Programa servidor - srv_03.c

   O cliente envia um código de operação e nome do utilizador ao servidor
   e este escreve no ecrã "<Username> has requested operation <opcode>".
   O servidor termina quando receber 'opcode' igual a zero.
*/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <sys/file.h>
#include <signal.h>
#include <errno.h>
#include <string.h>

#define MAX_NAME_LEN 20

int main(void)
{
 int   fd, fd_dummy;
 char  name[MAX_NAME_LEN];
 int opcode;

 if (mkfifo("/tmp/requests",0660)<0)
  if (errno==EEXIST) printf("FIFO '/tmp/requests' already exists\n");
  else printf("Can't create FIFO\n");
 else printf("FIFO '/tmp/requests' sucessfully created\n");

 if ((fd=open("/tmp/requests",O_RDONLY)) !=-1)
  printf("FIFO '/tmp/requests' openned in READONLY mode\n");

 if ((fd_dummy=open("/tmp/requests",O_WRONLY)) !=-1)
  printf("FIFO '/tmp/requests' openned in WRITEONLY mode\n");

 do
  {
    read(fd,&opcode,sizeof(int));
    if (opcode!=0) {
     read(fd,name,MAX_NAME_LEN);
     printf("%s has requested operation %d\n",name,opcode);
    }
  } while (opcode!=0);

  close(fd);
  close(fd_dummy);
  if (unlink("/tmp/requests")<0)
   printf("Error when destroying FIFO '/tmp/requests'\n");
  else
   printf("FIFO '/tmp/requests' has been destroyed\n");
  exit(0);
}
```

```c
/* SISTEMAS OPERATIVOS
   Arquitectura cliente-servidor

   Programa cliente  - cli_03.c

   O cliente envia um código de operação e nome do utilizador ao servidor
   e este escreve no ecrã "<Username> has requested operation <opcode>".
*/

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

#define MAX_MSG_LEN 20

int main(int argc, char *argv[])
{
 int   fd, opcode;

 if (argc!=2 && argc!=3) {
  printf("Usage: cli_03 <opcode> <username> OR cli_03 0\n");
  exit(1);
 }

 fd=open("/tmp/requests",O_WRONLY);
 if (fd == -1) {
  printf("Oops !!! Service is closed !!!\n");
  exit(1);
 }

 printf("FIFO 'requests' openned in WRITEONLY mode\n");

 // QUAL É O PROBLEMA DESTE CÓDIGO ?
 // (considerar a existência de múltiplos clientes)
 // A FAZER: implementar a solução correcta
 opcode=atoi(argv[1]);
 write(fd,&opcode,sizeof(int));
 if (opcode!=0) {
  write(fd,argv[2],strlen(argv[2])+1);
 }
 close(fd);
 return 0;
}
```