```c
#include "hello.h"
void gest_sigpipe(int sig){
  fprintf(stderr,"\n%s:serveur_déconnecté\n",prgname);
  exit(1);
}

int main(int argc, char** argv)
{
  prgname = argv[0];
  int statut;

  /* vérification des arguments */
  if ( argc!=3 ) {
      fprintf(stderr,"%s:usage %s serveur port\n",prgname,prgname);
      exit(1);
  }
  char* namesvr = argv[1];
  char* service = argv[2];
  /* création du SAP du serveur */
  socklen_t       svrSAPlen;
  struct sockaddr svrSAP;
  getTCPsap(&svrSAP,&svrSAPlen,namesvr,service);

  /* connexion au serveur */
  int cx;
  if ( (cx=socket(AF_INET,SOCK_STREAM,0))==-1 ) {
    fprintf(stderr,"%s: pb socket: %s\n",argv[0],strerror(errno));
    exit(1);
  }
  if ( connect(cx,&svrSAP,svrSAPlen)==-1 ) {
    fprintf(stderr,"%s: pb connect: %s\n",argv[0],strerror(errno));
     exit(1);
   }
  /*E2Q2
  statut = write(cx,"OK",3);
  close(cx);exit(0);
  */
  char PDU[100];
  /* Dialogue état début */
  statut = lire_PDU(PDU,cx );
  if ( statut!='H' ) goto error;
printf("client: reçu \"%s\" :", PDU); fflush(stdout);
/*E2Q1
signal(SIGPIPE,gest_sigpipe);*/
 statut = write(cx,"O",1);
  statut = write(cx,"K",2);
printf(" envoyé \"OK\" :"); fflush(stdout);

  printf(" reçu \"%s\" : quitte\n", PDU);

  /* Terminaison état fin */
```

```c
 /* Dialogue état att-fin */
  statut = lire_PDU(PDU,cx );
  if ( statut!='F' ) goto error;
printf(" reçu \"%s\" : quitte\n", PDU);

  /* Terminaison état fin */
  close(cx);
  return 0;

error:
  fprintf(stderr,"%s: message de type %c est inattendu\n",
          prgname,statut);
  return 1;
}
```

```c
#include "hello.h"
void gest_sigpipe(int sig){
  fprintf(stderr,"%s:client_déconnecté\n",prgname);
  exit(1);
}

int main(int argc, char** argv)
{
  prgname = argv[0];
  int statut;

  /* vérification des arguments */
  if ( argc!=2 ) {
      fprintf(stderr,"%s:usage: %s port\n",prgname,prgname);
      exit(1);
  }
  char* service = argv[1];

  /* création du SAP des clients */
  socklen_t       cltsSAPlen;
  struct sockaddr cltsSAP;
  getTCPsap(&cltsSAP,&cltsSAPlen,NULL,service);
  /* création de l'automate de connexion */
  int sock;
  if ( (sock=socket(AF_INET,SOCK_STREAM,0))==-1 ) {
    fprintf(stderr,"%s: pb socket: %s\n",argv[0],strerror(errno));
    exit(1);
  }
  if ( bind(sock,&cltsSAP,cltsSAPlen)<0 ) {
    fprintf(stderr,"%s: pb bind: %s\n",argv[0],strerror(errno));
     exit(1);
  }
  if ( listen(sock,100)!=0 ) {
     fprintf(stderr,"%s: pb listen:%s\n",argv[0],strerror(errno));
      exit(1);
  }
```

```c
while (1) {
    int cx;
    struct sockaddr cltSAP;
    socklen_t       cltSAPlen=sizeof(cltSAPlen);

    /* creation du flux de communication (cx) */
    if ( (cx=accept(sock,&cltsSAP,&cltsSAPlen))==-1 ) {
      fprintf(stderr,"%s: pb accept : sock=%d :
%s\n",argv[0],sock,strerror(errno));
      exit(1);
    }

    char PDU[100];
    /* Dialogue état début */

    sprintf(PDU,"HELLO");
    statut = write(cx,PDU,6);
    if ( statut==-1 ) { close(cx); continue; }
  printf("serveur: envoyé \"HELLO\" : "); fflush(stdout);
    /*E2Q1:close(cx);continue;*/
    /* Dialogue état att-ok */
    statut = lire_PDU(PDU,cx);
    if ( statut!='O' ) goto error;
  printf("reçu \"%s\" :", PDU); fflush(stdout);
/*E2Q2
  signal(SIGPIPE,gest_sigpipe);*/


  sprintf(PDU,"FIN");
    statut = write(cx,PDU,4);

    if ( statut==-1 ) { close(cx); continue; }
  printf(" envoyé \"FIN\" : quitte\n");


     /* Terminaison du dialogue (état fin) */
    close(cx);
    continue;

  error:
    close(cx);
    fprintf(stderr,"%s: message de type %c est inattendu\n",
            prgname,statut);
  }


   close(sock);
  return 0;
}
```

```c
if ( argc<2 ) {
      fprintf(stderr,"%s:usage: %s port\n",prgname,prgname);
      exit(1);
  }
/*E3Q2*/srv
  int duree = argc==2 ? 0: atoi(argv [2])*100000;
// pid_t PID1 = fork();
  if (PID1!=0){ pid_t PID2=fork();}
//while(1)
    /*E2Q1:close(cx);continue;*/
    /* Dialogue état att-ok */
    statut = lire_PDU(PDU,cx);
    if ( statut!='O' ) goto error;
  printf("reçu \"%s\" :", PDU); fflush(stdout);
sprintf(PDU,"FIN");
  /*E3Q2*/
  if(duree!=0)usleep(duree);
pid_t pid = fork();
            if (pid == 0) {
                  char PDU[100];
                  /* Dialogue état début */
                  statut = write(cx,"HELLO",6);
                  if ( statut==-1 ) { close(cx); continue; }
                  printf("serveur: envoyé \"HELLO\" : ");
fflush(stdout);
                  /* Dialogue état att-ok */
                  statut = lire_PDU(PDU,cx);
                  if ( statut!='O' ) goto error;
                  printf("reçu \"%s\" :", PDU); fflush(stdout);
                  if (tp != 0) {
                        usleep(tp);
                  }
                  statut = write(cx,"FIN",4);
                  if ( statut==-1 ) { close(cx); continue; }
                  printf(" envoyé \"FIN\" : quitte\n");
                  /* Terminaison du dialogue (état fin) */
                  close(cx);
                  exit(0);
error:          close(cx);
                  fprintf(stderr,"%s: message de type %c est
inattendu\n",
                                    prgname,statut);
                  exit(1);
            }

/*E3Q1*/clt
  int duree = argc==3 ? 0: atoi(argv [3])*100000;
 /*E3Q1*/
  if(duree!=0)usleep(duree);
```

```c
void getTCPsap(struct sockaddr* sap, socklen_t*saplen,
               const char *host, const char* port)
{int status;
  struct addrinfo hints,*found;
  memset(&hints, 0, sizeof(struct addrinfo));
  hints.ai_flags    = host!=0 ? 0 : AI_PASSIVE;
  hints.ai_family   = AF_INET;
  hints.ai_socktype = SOCK_STREAM;
  status=getaddrinfo(host, port,&hints,&found);
  if ( status!=0 ) {
      fprintf(stderr,"%s: pb getaddrinfo :
%s\n",prgname,gai_strerror(status));
      exit(1);
  }
  *sap    = *found->ai_addr;
  *saplen =  found->ai_addrlen;
#if 0 // debug
  struct addrinfo *p;
  for ( p=found ; p!=0 ; p = p->ai_next ) {
      struct sockaddr_in *sap =  p->ai_addr;
      short port = ntohs(sap->sin_port);
      printf("prot=%d/%08x len=%d sap=%p addr=%x port=%d
%d\n",p->ai_protocol,p->ai_protocol,
              p->ai_addrlen,p->ai_addr,
              sap->sin_addr.s_addr, sap->sin_port, port);
  }#endif
}
int lire_data(char*buf, int cx, int n)

{    int i,statut;
    for (i=0 ; i<n ; i++) {
      if ( (statut=read(cx,buf+i,1))!=1 ) {
        if ( ! silence )
            fprintf(stderr,"%s: pb lecture : %s\n",prgname,
              statut<0 ? strerror(errno) : "fin inattendue");
      return -1;
     }}
return n;
}
```

```c
int lire_PDU(char*buf, int cx)
{
    int nb;
    if ( lire_data(buf,cx,1)!=1 ) return '?';
    if ( *buf=='H' ) {
        nb =  lire_data(buf+1,cx,5);
        return nb==5 && strcmp(buf,"HELLO")==0 ? 'H' : 'e';
    } else if ( *buf=='O' ) {
        nb =  lire_data(buf+1,cx,2);
        return nb==2 && strcmp(buf,"OK")==0 ? 'O' : 'e';
    } else if ( *buf=='F' ) {
        nb =  lire_data(buf+1,cx,3);

        return nb==3 && strcmp(buf,"FIN")==0 ? 'F' : 'e';

    }

    return '?'

}#endif // FILE_HELLO_H
int lire_data(char*buf, int cx, int n)
{
      int i,statut;
      for (i=0 ; i<n ; i++) {
            if ( to>0 ) {
                  struct pollfd fds = { cx, POLLIN | POLLOUT, 0 };
                  if ( (statut=poll(&fds,1,300))<=0 ) {
                        if ( ! silence )
                              fprintf(stderr,"%s: pb lecture :
%s\n",prgname,
                                                    statut<0 ? strerror(errno) :
"timeout");
                        return -i;
                  }
            }
            if ( (statut=read(cx,buf+i,1))!=1 ) {
                  if ( ! silence )
                        fprintf(stderr,"%s: pb lecture : %s\n",prgname,
                                    statut<0 ? strerror(errno) : "fin
inattendue");
                  return -i;
            }
      }
      return n;
}
```