```c
/* Lab 4 - client.c */

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <sys/time.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>

#define PORT 40001    /* the port client will be connecting to */
#define MAXDATASIZE 1000 /* max number of bytes we can get at once */

int main(int argc, char *argv[]) {
  int sockfd;
  char inbuf[MAXDATASIZE], outbuf[MAXDATASIZE];
  struct hostent *he;
  struct sockaddr_in their_addr; /* connector's address information */
  int numInLines = 0, numOutLines = 0, maxlinesize = 1, totalInChars = 0, totalOutChars = 0;
  FILE *rsock, *wsock;

  /* Variáveis para o select() */
  fd_set master;
  fd_set temp;

  int done = 0;
  int fdmax;

  if (argc != 2) {
    fprintf(stderr,"usage: client hostname\n");
    exit(1);
  }

  /* Limpa os conjuntos de file descriptors master e read_fds */
  FD_ZERO(&master);
  FD_ZERO(&temp);

  if ((he=gethostbyname(argv[1])) == NULL) {  /* get the host info */
    perror("gethostbyname");
    exit(1);
  }
  if ((sockfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
    perror("socket");
    exit(1);
  }

  their_addr.sin_family = AF_INET;         /* host byte order */
  their_addr.sin_port = htons(PORT);     /* short, network byte order */
  their_addr.sin_addr = *((struct in_addr *)he->h_addr);
  bzero(&(their_addr.sin_zero), 8);        /* zero the rest of the struct */

  if (connect(sockfd, (struct sockaddr *)&their_addr, sizeof(struct sockaddr)) == -1) {
    perror("connect");
    exit(1);
  }
  if ((rsock = fdopen(sockfd, "r")) == NULL) {
    perror("fdopen");
    exit(1);
  }
  if ((wsock = fdopen(sockfd, "w")) == NULL) {
    perror("fdopen");
    exit(1);
  }

  /* Seta modo de buffer */
  setvbuf(wsock, NULL, _IOLBF, 0);
  setlinebuf(rsock);
  setlinebuf(stdout);
  setlinebuf(stdin);

  FD_SET(sockfd, &master);
  FD_SET(STDIN_FILENO, &master);
  fdmax = sockfd;

  if (fgets(inbuf, MAXDATASIZE, rsock) == NULL) {
    perror("fgets");
    exit(1);
  }
  fflush(rsock);
  fprintf(stderr, "Received: %s", inbuf);
  struct timeval start;
  gettimeofday( &start, NULL );

  while (1) {
    temp = master;              /* Salvando grupo de fds. */
    /* Checando se há algo no stdin ou no sockfd */
    if (select(fdmax+1, &temp, NULL, NULL, NULL) < 0) {
      perror("select");
      exit(1);
    }
    if (FD_ISSET(sockfd, &temp)) { /* Temos algo pra ler do servidor */
      if (fgets(inbuf, MAXDATASIZE, rsock) == NULL) { /* Já recebemos tudo */
        break;
      }
      fflush(rsock);
      numInLines++;
      totalInChars += strlen(inbuf);
      printf("%s", inbuf);
    }
    if (FD_ISSET(STDIN_FILENO, &temp)) { /* Lendo da entrada padrão e mandar pro servidor */
      if (fgets(outbuf, MAXDATASIZE, stdin) != NULL) {
        if (fputs(outbuf, wsock) == EOF) {
          perror("send");
          exit(1);
        }
        numOutLines++;
        maxlinesize = maxlinesize > strlen(outbuf) ? maxlinesize : strlen(outbuf);
        totalOutChars += strlen(outbuf);
      }
      else {
        if (!done) {
          shutdown(sockfd, SHUT_WR); /* Já mandamos tudo. */
          fprintf(stderr, "Numero de linhas enviadas: %d\n", numOutLines);
          fprintf(stderr, "Numero de caracteres na maior linha: %d\n", maxlinesize - 1);
          fprintf(stderr, "Total de caracteres enviados: %d\n", totalOutChars);
          done = 1;
        }
      }
    }
  }

  struct timeval end;
  gettimeofday( &end, NULL );
  double time_elapsed = ( (double)( end.tv_usec - start.tv_usec ) ) /
    1.0e+6 + ( (double)( end.tv_sec - start.tv_sec ) );
  fprintf( stderr, "Tempo total de transferencia: %lf s\n", time_elapsed );
  fprintf(stderr, "Numero de linhas recebidas: %d\n", numInLines);
  fprintf(stderr, "Total de caracteres recebidos: %d\n", totalInChars);

  fclose(rsock);
  close(sockfd);
  return 0;

}
```