MC833 - 2s2017
Bruno Orsi Berton **RA 150573 - Turma B**
Fábio Takahashi Tanniguchi **RA 145980 - Turma A**

servidor.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <time.h>
#include <unistd.h>

#define LISTENQ 10
#define MAXDATASIZE 100

int main (int argc, char **argv) {
   int    listenfd, connfd, peeraddr_len;
   struct sockaddr_in servaddr, peeraddr;
   char   buf[MAXDATASIZE];
   time_t ticks;

   // cria um socket TCP
   if ((listenfd = socket(AF_INET, SOCK_STREAM, 0)) == -1) {
      perror("socket");
      exit(1);
   }

   // configura os parâmetros da conexão
   bzero(&servaddr, sizeof(servaddr));
   servaddr.sin_family      = AF_INET;
   servaddr.sin_addr.s_addr = INADDR_ANY;
   servaddr.sin_port        = htons(13000);

   // faz o bind do socket TCP com o host:porta escolhidos
   if (bind(listenfd, (struct sockaddr *)&servaddr, sizeof(servaddr)) == -1) {
      perror("bind");
      exit(1);
   }

   // ativa a socket para começar a receber conexões
   if (listen(listenfd, LISTENQ) == -1) {
      perror("listen");
      exit(1);
   }

   // espera por conexões de clientes indefinidamente
   for ( ; ; ) {
```

```
      // aceita as conexões
      if ((connfd = accept(listenfd, (struct sockaddr *) NULL, NULL)) == -1 ) {
         perror("accept");
         exit(1);
      }

      ticks = time(NULL);
      snprintf(buf, sizeof(buf), "%.24s\r\n", ctime(&ticks));
      // escreve no socket para que o cliente receba a mensagem
      write(connfd, buf, strlen(buf));

      printf("Recebi uma conexão\n");

      // imprime dados do socket do cliente
      peeraddr_len = sizeof(struct sockaddr);
      if (getpeername(connfd, (struct sockaddr *) &peeraddr, &peeraddr_len) == -1) {
         perror("getpeername() failed");
         return -1;
      }
      printf("IP address do cliente: %s\n", inet_ntoa(peeraddr.sin_addr));
      printf("Porta do cliente: %d\n", (int) ntohs(peeraddr.sin_port));

      close(connfd);
   }

   return(0);
}
```

cliente.c

```
#include <sys/socket.h>
#include <sys/types.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <stdio.h>
#include <netdb.h>
#include <string.h>
#include <errno.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>

#define MAXLINE 4096

int main(int argc, char **argv) {
   int    sockfd, n, servaddr_len;
   char   recvline[MAXLINE + 1];
   char   error[MAXLINE + 1];
   struct sockaddr_in servaddr;
   struct hostent *hp;

   // verifica se o host foi passado como parametro
   if (argc != 2) {
      strcpy(error,"uso: ");
```

```
   strcat(error,argv[0]);
   strcat(error," <IPaddress>");
   perror(error);
   exit(1);
}

// cria um socket TCP
if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
   perror("socket error");
   exit(1);
}

// recupera o IP através do host passado por parametro
     hp = gethostbyname(argv[1]);
     if (!hp) {
            fprintf(stderr, "simplex-talk: unknown host: %s\n", argv[1]);
            exit(1);
     }

// configura os parâmetros da conexão
bzero(&servaddr, sizeof(servaddr));
servaddr.sin_family = AF_INET;
servaddr.sin_addr.s_addr = inet_addr(argv[1]);
servaddr.sin_port   = htons(13000);

if (inet_pton(AF_INET, argv[1], &servaddr.sin_addr) <= 0) {
   perror("inet_pton error");
   exit(1);
}

// abre a conexão com o servidor
if (connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr)) < 0) {
   perror("connect error");
   exit(1);
}

// le o que foi recebido através do socket e imprime o conteúdo
while ( (n = read(sockfd, recvline, MAXLINE)) > 0) {
   recvline[n] = 0;
   if (fputs(recvline, stdout) == EOF) {
      perror("fputs error");
      exit(1);
   }
}

if (n < 0) {
   perror("read error");
   exit(1);
}

// imprime dados do socket
servaddr_len = sizeof(struct sockaddr_in);
if (getsockname(sockfd, (struct sockaddr *) &servaddr, &servaddr_len) == -1) {
  perror("getsockname() failed");
```

```
      return -1;
    }
    printf("IP address do socket: %s\n", inet_ntoa(servaddr.sin_addr));
    printf("Client port do socket: %d\n", (int) ntohs(servaddr.sin_port));

    exit(0);
}
```