Atividade 6 - Implementacao de Aplicacao Cliente/Servidor similar a telnet e concorrente com TCP- Códigos

Gabriel Hidasy Rezende

November 1, 2015

```
 //Q1-2 cliente
 #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>
#include "netutils.h"
#define MAXLINE 4096

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

   if (argc != 3){
      strcpy(error,"uso: ");
      strcat(error,argv[0]);
      strcat(error," <IPaddress> <Port Number>");
      perror(error);
      exit(1);
   }
   int pnumber;
```

```
   sscanf(argv[2],"%d",&pnumber);
   /* Abre o socket */
   sockfd = Socket(AF_INET, SOCK_STREAM, 0);

   /* Prepara as estruturas que serao usadas */
   bzero(&servaddr, sizeof(servaddr));
   servaddr.sin_family = AF_INET;
   servaddr.sin_port = htons(pnumber);

   /* Aqui e criada a estrutura que sera usada para a conexao */
   Inet_pton(AF_INET, argv[1], &servaddr.sin_addr);

   /* A conexao e feita */
   Connect(sockfd, (struct sockaddr *) &servaddr, sizeof(servaddr));
   printf("Connected to %s:%d\n",argv[1],pnumber);

   char command[1025];
   while (1) {
     /* Le a linha de comando */
     n = scanf(" %[^\n]",command);
     if (n == 0) {
       break;
     }
     /* Envia o comando */
     Write(sockfd,command,strlen(command));
     /* Recebe a resposta */
     n = Read(sockfd, recvline, MAXLINE);
     if (n == 0) {
       break;
     }
     recvline[n] = 0;
     printf("%s\n",recvline);
   }
   close(sockfd);
   exit(0);
}
```

```
  //Q1-2 servidor
 #include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <string.h>
#include <netdb.h>
#include <sys/types.h>
```

```c
#include <netinet/in.h>
#include <sys/socket.h>
#include <time.h>
#include <unistd.h>
#include <arpa/inet.h>
#include "netutils.h"

#define LISTENQ 10
#define MAXDATASIZE 4096

void deal_with_client(int connfd)
{
  char command[MAXDATASIZE];
  int n;
  while(1) {
    n = Read(connfd,command,MAXDATASIZE-1);
    if (n == 0) {
      break;
    }
    command[n] = 0;
    //Thats a security flaw, try to chroot this server
    system(command);
    Write(connfd,command,n);
  }
}

int main (int argc, char **argv)
{
  int   listenfd, connfd;
  struct sockaddr_in servaddr;
  int pnumber = 12344;
  if (argc == 2) {
    sscanf(argv[1],"%d",&pnumber);
   }
   /* No trecho abaixo um socket e inicializado */
   listenfd = Socket(AF_INET, SOCK_STREAM, 0);

   /* Estruturas sao preparadas */
   bzero(&servaddr, sizeof(servaddr));
   servaddr.sin_family    = AF_INET;
   servaddr.sin_addr.s_addr = htonl(INADDR_ANY);
   servaddr.sin_port      = htons(pnumber);

   /* A porta e reservada */
   Bind(listenfd,(struct sockaddr *) &servaddr, sizeof(servaddr));
```

```
   /* O servidor passa a receber conexoes */
   Listen(listenfd,LISTENQ);

   for ( ; ; ) {
     struct sockaddr_in client;
     connfd = Accept(listenfd,&client);
     char client_address[128];
     int client_port = ntohs(client.sin_port);
     inet_ntop(AF_INET,&client.sin_addr.s_addr,client_address,128);
     printf("Serving client %s:%d\n",client_address,client_port);

      int chid = fork();
      if (chid == 0) {
   deal_with_client(connfd);
   close(connfd);
   exit(0);
      } else {
   close(connfd);
      }
   }
   return(0);
}
```

```
//Netutils.c
#include "netutils.h"
int Socket(int family, int type, int flags)
{
  int sockfd;
  if ((sockfd = socket(family, type, flags)) < 0) {
    perror("socket");
    exit(1);
  } else
    return sockfd;
}
void Bind(int sockfd, struct sockaddr *servaddr, int size)
{
  if (bind(sockfd,
      servaddr, size) == -1) {
    perror("bind");
    exit(1);
  }
}
```

```
void Listen(int listenfd, int flags)
{
  if (listen(listenfd, flags) == -1) {
      perror("listen");
      exit(1);
   }
}

int Read(int connfd, char *command, int size) {
  int n;
  n = read(connfd,command,size);
  if (n < 0) {
    perror("read");
  }
  return n;
}

int Accept(int listenfd, struct sockaddr_in *client)
{
  int connfd;
  socklen_t client_size = sizeof(*client);
  if ((connfd = accept(listenfd, (struct sockaddr *) client,
      &client_size)) == -1 ) {
         perror("accept");
         exit(1);
      }
  return connfd;
}
void Write(int connfd, char *command, int n)
{
  if(write(connfd, command, n) < 0) {
    perror("write");
    exit(1);
  }
}
void Connect(int sockfd, struct sockaddr *servaddr, int size)
{
  if (connect(sockfd, servaddr, size) < 0) {
      perror("connect");
      exit(1);
   }
}
void Inet_pton(int family, char *in, struct in_addr *servaddr)
{
  if (inet_pton(family, in, servaddr) <= 0) {
```

```
    perror("inet_pton error");
    exit(1);
  }
}
```

```
//netutils.h
#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>

/* Poderia ter simplificado algumas chamadas, eliminando */
/* os int size fazendo os mesmos dentro dos wrappers, mas */
/* do jeito que fiz eles sao compativeis o bastante para */
/* serem usados como drop-in replacements para as    */
/*funcoes de baixo nivel                             */

int Socket(int family, int type, int flags);
void Bind(int sockfd, struct sockaddr *servaddr, int size);
void Listen(int listenfd, int flags);
int Read(int connfd, char *command, int size);
int Accept(int listenfd, struct sockaddr_in *client);
void Write(int connfd, char *command, int n);
void Connect(int sockfd, struct sockaddr *servaddr, int size);
void Inet_pton(int type, char *in, struct in_addr *servaddr);
```