ΠΠΣ
Εργαστηριακές Σημειώσεις - Υποδοχές

ΕΡΓΑΣΤΗΡΙΟ Λ.Σ.

Εαρινό 2017

1 Το αρχείο server.c

Listing 1: file.c

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

#define PORT 4444  //change this
#define BUF_SIZE 2000
#define CLADDR_LEN 100

void * receiveMessage(void * socket) {
  int sockfd, ret;
  char buffer[BUF_SIZE];
  sockfd = *((int*)socket);
  memset(buffer, 0, BUF_SIZE);
  for (;;) {
    ret = recvfrom(sockfd, buffer, BUF_SIZE, 0, NULL, NULL);
    if (ret < 0) {
      printf("Error receiving data!\n");
    } else {
      printf("client: ");
      fputs(buffer, stdout);
//printf("\n");
    }
  }
}

int main() {
  struct sockaddr_in addr, cl_addr;
  int sockfd, len, ret, newsockfd;
  int yes=1;
  char buffer[BUF_SIZE];
  char clientAddr[CLADDR_LEN];
  pthread_t rThread;

  sockfd = socket(AF_INET, SOCK_STREAM, 0); //create tcp socket
  if (sockfd < 0) {
    printf("Error creating socket!\n");
    exit(1);
  }
  //Permits multiple sockets to be bound to an identical socket
       address
  if (setsockopt(sockfd, SOL_SOCKET, SO_REUSEADDR, &yes, sizeof
      (int)) == -1) {
    perror("setsockopt");
    exit(1);
  }
  printf("Socket created...\n");

  memset(&addr, 0, sizeof(addr));
  addr.sin_family = AF_INET;
  addr.sin_addr.s_addr = INADDR_ANY;
  addr.sin_port = PORT;

  ret = bind(sockfd, (struct sockaddr *) &addr, sizeof(addr));
  if (ret < 0) {
    printf("Error binding!\n");
    exit(1);
  }
  printf("Binding done...\n");

  printf("Waiting for a connection...\n");
  listen(sockfd, 5);


  len = sizeof(struct sockaddr_in);
  newsockfd = accept(sockfd, (struct sockaddr*)&cl_addr, (socklen_t*)&len);
  if (newsockfd < 0) {
    printf("Error accepting connection!\n");
    exit(1);
  }
  //converts the network address structure into a character string
  inet_ntop(AF_INET, &(cl_addr.sin_addr), clientAddr, CLADDR_LEN);
  printf("Connection accepted from %s...\n", clientAddr);

  memset(buffer, 0, BUF_SIZE);
  printf("Enter your messages one by one and press return key!\n");

//creating a new thread for receiving messages from the client
  ret = pthread_create(&rThread, NULL, receiveMessage, &newsockfd);
  if (ret) {
    printf("ERROR: Return Code from pthread_create() is %d\n", ret);
    exit(1);
  }

  while (fgets(buffer, BUF_SIZE, stdin) != NULL) {
    ret = sendto(newsockfd, buffer, BUF_SIZE, 0, (struct sockaddr *) &cl_addr, len);
    if (ret < 0) {
      printf("Error sending data!\n");
      exit(1);
    }
  }

  close(newsockfd);
  close(sockfd);

  pthread_exit(NULL);
  return 0;
}
```

2 Το αρχείο client.c

Listing 2: file.c

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

#define PORT 4444  //change this
#define BUF_SIZE 2000

void * receiveMessage(void * socket) {
 int sockfd, ret;
 char buffer[BUF_SIZE];
 sockfd = *((int*)socket);
 memset(buffer, 0, BUF_SIZE);
 for (;;) {
  ret = recvfrom(sockfd, buffer, BUF_SIZE, 0, NULL, NULL);
  if (ret < 0) {
   printf("Error receiving data!\n");
  } else {
   printf("server: ");
   fputs(buffer, stdout);
  }
 }
}

int main(int argc, char**argv) {
 struct sockaddr_in addr;
 int sockfd, ret;
 char buffer[BUF_SIZE];
 char * serverAddr;
 pthread_t rThread;

 if (argc < 2) {
  printf("usage: client < ip address >\n");
  exit(1);
 }

 serverAddr = argv[1];

 sockfd = socket(AF_INET, SOCK_STREAM, 0);
 if (sockfd < 0) {
  printf("Error creating socket!\n");
  exit(1);
 }
 printf("Socket created...\n");

 memset(&addr, 0, sizeof(addr));
 addr.sin_family = AF_INET;
 addr.sin_addr.s_addr = inet_addr(serverAddr);
 addr.sin_port = PORT;

 ret = connect(sockfd, (struct sockaddr *) &addr, sizeof(addr))
     ;
 if (ret < 0) {
  printf("Error connecting to the server!\n");
  exit(1);
 }
 printf("Connected to the server...\n");

 memset(buffer, 0, BUF_SIZE);
 printf("Enter your messages one by one and press return key!\n
     ");

 //creating a new thread for receiving messages from the server
 ret = pthread_create(&rThread, NULL, receiveMessage, &sockfd);
 if (ret) {
  printf("ERROR: Return Code from pthread_create() is %d\n",
      ret);
  exit(1);
 }

 while (fgets(buffer, BUF_SIZE, stdin) != NULL) {
  ret = sendto(sockfd, buffer, BUF_SIZE, 0, (struct sockaddr *)
      &addr, sizeof(addr));
  if (ret < 0) {
   printf("Error sending data!\n\t-%s", buffer);
  }
 }

 close(sockfd);
 pthread_exit(NULL);

 return 0;
}
```