Esame Sistemi di Elaborazione dell'informazione. Parte B 26-01-2009

Potete tenere libri e appunti. Il tempo per l'esecuzione della prova è 1h30min

Esercizio 1 (15 punti) Con riferimento al server che segue (1) fornite lo pseudocodice delle funzioni failmess(), tcp_passive_open(), tcp_accept(), specificando le chiamate socket utilizzate (2) discutere cosa fa il server (2) fornite lo pseudocodice di un client adeguato (3) discutete come si possa implementare il server con la chiamata select(), immaginando che l'indirizzo IP del client sia noto a priori.

```
/*  gli include sono stati  omessi */
#define PORTNO  6789          /*  porta da usare */
int   port = PORTNO;
#define BUFF_LEN 200
char buff[BUFF_LEN];
int buf_len = BUFF_LEN-1;
char *progname;
main(int argc, char **argv)
{   int    port_sk,  client_sk;
    int    len;
    char *errmess;
    progname = argv[0];
    if ( parse_network_args( &argc, argv, NULL, &port, &errmess ) != 0 )
        failmess(errmess);
    port_sk = tcp_passive_open(port); /* set localhost for server, negative result on failure    */
    if ( port_sk < 0 ) { perror("socket"); exit(1); }
    printf("start up complete\n");
    client_sk = tcp_accept(port_sk); /*  attesa connessione del client*/
    close(port_sk);
    for(;;) {  /*  parliamo al client  */
              len = read(client_sk,buff,buf_len); /*  attesa messaggio del client  */
        if (len == 0) {
            printf("il client ha terminato la connessione\n"); break;}
        buff[len] = '\0';
        printf("client says: %s\n",buff);
      /*  e' il nostro turno  */
        printf("messaggio: ");
        if ( gets(buff) == NULL ) {    /* utente ha digitato EOF  */
            close(client_sk);
            printf("arrivederci\n");break;}
        write(client_sk,buff,strlen(buff));
    } exit(0);}
```

Esercizio 2 (8 punti) Spiegate in dettaglio la sessione che segue.

(A) E' un colloquio telnet o http? A cosa puo' servire? Definite il significato e lo scopi di tutti i campi dello header.

```
% telnet www.cosp.unimi.it 80
Trying 159.149.64.12...
Connected to www.cosp.unimi.it.
Escape character is '^]'.
$> HEAD /offerta_didattica/96.htm HTTP/1.0

HTTP/1.1 200 OK
Date: Tue, 24 Jan 2009 18:01:05 GMT
Server: Apache/1.3.26 (Unix) mod_perl/1.24_01 mod_ssl/2.8.10 OpenSSL/0.9.5a
Last-Modified: Tue, 18 Jan 2009 14:38:31 GMT
ETag: "101e6a1-cd-3e5237be"
Accept-Ranges: bytes
Content-Length: 205
Connection: close
Content-Type: text/html; charset=iso-8859-1

% Connection closed by foreign host.
```

(B) Quale sarebbe il risultato della richiesta:

```
GET /offerta_didattica/96.htm HTTP/1.0
If-Modified-Since: Mon, 26 Jan 2009 14:38:31 GMT
```

Esercizio 3 (7 punti) Bob deve scrivere una email ad Alice. Il suo host è configurato come segue

- IP address: 192.45.56.127
- Name server: 192.47.56.2
- SMTP server: mail.server.org

L'indirizzo di Alice è alice@wonderland.org, quello di Bob è bob@realword.org.

1. Perchè il client di posta di Bob possa collegarsi al server SMTP, occorre tradurre il nome del server in indirizzo IP usando il DNS. Elencate e spiegate i messaggi che vengono scambiati, supponendo che SOLO il name server del dominio server.org conosca l'indirizzo corrispondente a mail.server.org, e che il suo valore sia 139.49.111.22.
2. Dopo la traduzione il client di posta può inviare la posta. Mostrate i segmenti scambiati mettendo in evidenza gli header TCP (solo gli indirizzi) e il contenuto (messaggi SMTP) .

Soluzione esercizio 1

```
void failmess(char *mess)
{
    fprintf( stderr, "%s: %s\n", progname, mess );
    exit(1);
}
-----------------------------------------------------------
tcp_passive_open():
creazione socket
compilazione strutture addr
bind-listen
-----------------------------------------------
int  tcp_passive_open(portno)
    int    portno;
{
    int     sd, code;
    struct  sockaddr_in bind_addr;
    bind_addr.sin_family = AF_INET;
    bind_addr.sin_addr.s_addr = 0;    /*  0.0.0.0  ==  this host  */
    bzero(bind_addr.sin_zero, 8);
    bind_addr.sin_port = portno;
    sd = socket(AF_INET, SOCK_STREAM,0);
    if ( sd < 0 ) return sd;
    code = bind(sd, &bind_addr, sizeof(bind_addr) );
    if ( code < 0 ) { close(sd); return code; }
    code = listen(sd, 1);
    if ( code < 0 ) { close(sd); return code; }
    return sd;
}
-----------------------------------------------
tcp_accept()
accept
-----------------------------------------------
int  tcp_accept(sock)
    int sock;
{
    int     sd;
    struct  sockaddr bind_addr;
    int len=sizeof(bind_addr);
    sd = accept(sock, &bind_addr, &len);
    return sd;
}

/***************************************/
/*      simple-client.c                */
/*      Alan Dix                       */
/*      client for a simple 'talk'     */
/*      application with strict        */
/*      turn-taking                    */
/***************************************/

#include <stdio.h>
#include <string.h>
#include "sock.h"


#define PORTNO  6789          /*  default port and host  */
#define HOST    "zeus"

char *host = HOST;
int   port = PORTNO;

#define BUFF_LEN 200
char buff[BUFF_LEN];
int buf_len = BUFF_LEN-1;     /*  allow room for terminating '\0'  */

char *progname;

void failmess(char *mess)
{
    fprintf( stderr, "%s: %s\n", progname, mess );
    exit(1);
}
```

```
main( int argc, char**argv )
{
    int  serv_sk, len;
    char *errmess;

    progname = argv[0];

    if ( parse_network_args( &argc, argv, &host, &port, &errmess ) != 0 )
        failmess(errmess);

  /*  request connection to server  */
    serv_sk = tcp_active_open(host,port);
        /*  waits for server to accept             */
        /*  returns negative result on failure     */
        /*  host is serverÕs machine               */
    if ( serv_sk < 0 ) { perror("socket"); exit(1); }
    printf("You can send now\n");

/*  talk to server  */
    for(;;) {
      /*  our turn first  */
        printf("speak: ");
        if ( gets(buff) == NULL ) {    /* user typed end of file  */
            close(serv_sk);
            printf("bye bye\n");
            break;
            }
        write(serv_sk,buff,strlen(buff));

      /*  wait for serverÕs message  */
        len = read(serv_sk,buff,buf_len);
        if (len == 0) {
            printf("server finished the conversation\n");
            break;
            }
        buff[len] = '\0';
        printf("server says: %s\n",buff);
    }
    exit(0);
```