

Socket

Esercizio 1

Si scriva un client in grado di connettersi ad un host e porta specificati, tramite il protocollo TCP/IP e di leggere un pacchetto di dati. Il pacchetto ricevuto deve essere interpretato come una sequenza di caratteri e deve essere stampato.

Per la lettura di un pacchetto, si allochi un buffer di dimensione ragionevole (1024 bytes) e si esegua una semplice lettura dal socket (`recv()`). Si provi il programma con un servizio TCP/IP noto, ad es. `mail.polito.it:smtp` (ovvero `130.192.3.80:25`)

Esercizio 2

Si scriva un time server per il protocollo TCP. Il time sever deve porsi in ascolto alla porta 37. Quando un client si connette il server deve restituire una stringa contenente il tempo di sistema e chiudere la connessione.

Esercizio 3

Si scriva un programma in grado, dato il nome di un server (es. www.polito.it) di restituire il corrispondente indirizzo.

Si usi la funzione `struct hostent *gethostbyname(const char *name);`

Si stampino tutte le informazioni contenute nella struct `hostent`

Esercizio 4

La IANA (<http://www.iana.org/>) definisce delle porte standard per certi servizi (telnet, email etc.). Tali definizioni sono elencate nel file `/etc/services`.

Si scriva un programma che dato il nome (o la porta) di un servizio e il protocollo relativo (“tcp” o “udp”) restituisca le informazioni su questo protocollo.

Si usino le funzioni `getservbyname()` e `getservbyport()`;

Esercizio 5

Si integrino le funzionalità sviluppate negli esercizi 3 e 4 per migliorare il programma dell’esercizio 1.

Esercizio 6

Si modifichi il time server in modo che invece di chiudere il socket dopo aver inviato il tempo, mantenga la connessione ed invii periodicamente (almeno ogni secondo circa) il tempo di sistema.

Si modifichi anche il client in modo che, una volta connesso resti in ascolto per sempre. Si ignorino inizialmente gli errori.

NB la funzione `select()` consente di

- specificare un timeout
- capire se ci sono richieste di connessioni pendenti (== lettura)

Funzioni utili

gethostbyname

```
struct hostent *gethostbyname(const char *name);

struct hostent {
    char    *h_name;        /* official name of host */
    char    **h_aliases;    /* alias list */
    int     h_addrtype;     /* host address type */
    int     h_length;       /* length of address */
    char    **h_addr_list; /* list of addresses from name server */
};
```

The members of this structure are:

- `h_name` Official name of the host.
- `h_aliases` A NULL-terminated array of alternate names for the host.
- `h_addrtype` The type of address being returned; usually `AF_INET`.
- `h_length` The length, in bytes, of the address.
- `h_addr_list` A NULL-terminated array of network addresses for the host. Host addresses are returned in network byte order.

Getservbyname, getservbyport

```
struct servent *getservbyname(const char *name, const char *proto);

struct servent *getservbyport(int port, const char *proto);

struct servent {
    char    *s_name;        /* official name of service */
    char    **s_aliases;    /* alias list */
    int     s_port;         /* port service resides at */
    char    *s_proto;       /* protocol to use */
};
```

The members of this structure are:

- `s_name` The official name of the service.
- `s_aliases` A zero terminated list of alternate names for the service.
- `s_port` The port number at which the service resides. Port numbers are returned in network byte order.
- `s_proto` The name of the protocol to use when

Select

Esempio

```
while(1){
    int addrlen = sizeof(struct sockaddr_in);
    int nsockets;
    fd_set readfds;
    fd_set writefds;
    fd_set exceptfds;
    struct timeval timeout;

    FD_ZERO(&readfds);
    FD_ZERO(&writefds);
    FD_ZERO(&exceptfds);

    FD_SET(s,&readfds);
    FD_SET(s,&writefds);
    FD_SET(s,&exceptfds);

    timeout.tv_sec = 1;
    timeout.tv_usec = 0;

    nsockets = select(s+1,&readfds,NULL,NULL,&timeout);
    if(nsockets>0){
        s1 = accept(s, (struct sockaddr *) &caddr, &addrlen);
        /* do something with the socket */
        close(s1);
        printf("T");
        fflush(stdout);
    }else{
        printf(".");
        fflush(stdout);
    }
}
```