

Laboratorio reti AA 2007/2008

Dott. Matteo Roffilli

roffilli@csr.unibo.it

**Ricevimento in ufficio
dopo la lezione**

Laboratorio reti AA 2007/2008

Per esercitarvi fate SSH su:

`alfa.csr.unibo.it`

`si-tux00.csr.unibo.it`

`....`

`si-tux15.csr.unibo.it`

Lo username dovrebbe essere del tipo:

`STUDENTI/matteo.roffilli`

`STUDENTI\matteo.roffilli`

Se avete problemi di login contattate i tecnici

Eventuali variazioni di orario/giorno verranno comunicate in anticipo via mail.

Laboratorio reti AA 2007/2008

- **Marzo**
- 13 Intro,SSH,VI/VIM,GCC base
- 19 Richiami di C e Compilazione
- **Aprile**
- 3 Socket e Co.
- 10 Socket e Co. parte seconda
- 17 Client
- **23 Client parte II**

Union

Una union è una variabile che può contenere oggetti di tipo e dimensioni differenti che condividono lo stesso spazio di memoria.

E' utile quando si vuole accedere a medesime locazioni di memoria come attraverso una differente tipizzazione.

```
//definizione
```

```
typedef union  
{      int value;  
        char p_value[4];  
} byte_msg;
```

```
//dichiarazione
```

```
byte_msg header;
```

```
//utilizzo
```

```
header.p_value[0];  
header.value;
```

Gestione run-time degli errori

Quasi tutte le funzioni delle librerie del C sono in grado di individuare e riportare condizioni di errore, ed è una norma fondamentale di buona programmazione controllare sempre che le funzioni chiamate si siano concluse correttamente.

In genere le funzioni di libreria usano un valore speciale per indicare che c'è stato un errore. Di solito questo valore è -1 o un puntatore nullo o la costante EOF (a seconda della funzione); ma questo valore segnala solo che c'è stato un errore, non il tipo di errore.

Per riportare il tipo di errore il sistema usa la variabile globale **errno** definita nell'header **errno.h**.

Il valore di **errno** viene sempre impostato a zero all'avvio di un programma, gran parte delle funzioni di libreria impostano **errno** ad un valore diverso da zero in caso di errore. Il valore è invece indefinito in caso di successo, perché anche se una funzione ha successo, può chiamarne altre al suo interno che falliscono, modificando così **errno**.

Pertanto un valore non nullo di **errno** non è sintomo di errore (potrebbe essere il risultato di un errore precedente) e non lo si può usare per determinare quando o se una chiamata a funzione è fallita. La procedura da seguire è sempre quella di controllare **errno** immediatamente dopo aver verificato il fallimento della funzione attraverso il suo codice di ritorno.

perror e strerror

perror

```
#include <stdio.h>
void perror(const char *message)
```

Stampa il messaggio di errore relativo al valore corrente di `errno` sullo standard error; preceduto dalla stringa `message`.

strerror

```
#include <string.h>
char *strerror(int errnum)
```

Restituisce una stringa con il messaggio di errore relativo ad `errnum`.

La funzione ritorna il puntatore ad una stringa di errore.

Errori di rete 1

Di seguito sono raccolti i codici restituiti dalle funzioni di libreria attinenti ad errori che riguardano operazioni specifiche relative alla gestione dei socket e delle connessioni di rete.

ENOTSOCK Socket operation on non-socket. Si è tentata un'operazione su un file descriptor che non è un socket quando invece era richiesto un socket.

EMSGSIZE Message too long. Le dimensioni di un messaggio inviato su un socket sono eccedono la massima lunghezza supportata.

EPROTOTYPE Protocol wrong type for socket. Protocollo sbagliato per il socket. Il socket usato non supporta il protocollo di comunicazione richiesto.

ENOPROTOOPT Protocol not available. Protocollo non disponibile. Si è richiesta un'opzione per il socket non disponibile con il protocollo usato.

EPROTONOSUPPORT Protocol not supported. Protocollo non supportato. Il tipo di socket non supporta il protocollo richiesto (un probabile errore nella specificazione del protocollo).

ESOCKTNOSUPPORT Socket type not supported. Socket non supportato. Il tipo di socket scelto non è supportato.

EOPNOTSUPP Operation not supported on transport endpoint. L'operazione richiesta non è supportata. Alcune funzioni non hanno senso per tutti i tipi di socket, ed altre non sono implementate per tutti i protocolli di trasmissione. Questo errore quando un socket non supporta una particolare operazione, e costituisce una indicazione generica che il server non sa cosa fare per la chiamata effettuata.

EPFNOSUPPORT Protocol family not supported. Famiglia di protocolli non supportata. La famiglia di protocolli richiesta non è supportata.

Errori di rete 2

EAFNOSUPPORT Address family not supported by protocol. Famiglia di indirizzi non supportata. La famiglia di indirizzi richiesta non è supportata, o è inconsistente con il protocollo usato dal socket.

EADDRINUSE Address already in use. L'indirizzo del socket richiesto è già utilizzato (ad esempio si è richiesto il bind per una porta già in uso).

EADDRNOTAVAIL Cannot assign requested address. L'indirizzo richiesto non è disponibile (ad esempio si è cercato di dare al socket un nome che non corrisponde al nome della stazione locale).

ENETDOWN Network is down. L'operazione sul socket è fallita perché la rete è sconnessa.

ENETUNREACH Network is unreachable. L'operazione è fallita perché l'indirizzo richiesto è irraggiungibile (ad esempio la sottorete della stazione remota è irraggiungibile).

ENETRESET Network dropped connection because of reset. Una connessione è stata cancellata perché l'host remoto è caduto.

ECONNABORTED Software caused connection abort. Una connessione è stata abortita localmente.

ECONNRESET Connection reset by peer. Una connessione è stata chiusa per ragioni fuori dal controllo dell'host locale, come il riavvio di una macchina remota o un qualche errore non recuperabile sul protocollo.

ENOBUFS No buffer space available. Tutti i buffer per le operazioni di I/O del kernel sono occupati. In generale questo errore è sinonimo di ENOMEM, ma attiene alle funzioni di input/output. In caso di operazioni sulla rete si può ottenere questo errore invece dell'altro.

Errori di rete 3

EISCONN Transport endpoint is already connected. Si è tentato di connettere un socket che è già connesso.

ENOTCONN Transport endpoint is not connected. Il socket non è connesso a niente. Si ottiene questo errore quando si cerca di trasmettere dati su un socket senza avere specificato in precedenza la loro destinazione. Nel caso di socket senza connessione (ad esempio socket UDP) l'errore che si ottiene è **EDESTADDRREQ**.

EDESTADDRREQ Destination address required. Non c'è un indirizzo di destinazione predefinito per il socket. Si ottiene questo errore mandando dato su un socket senza connessione senza averne prima specificato una destinazione.

ESHUTDOWN Cannot send after transport endpoint shutdown. Il socket su cui si cerca di inviare dei dati ha avuto uno shutdown.

ETOOMANYREFS Too many references: cannot splice. La glibc dice ???

ETIMEDOUT Connection timed out. Un'operazione sul socket non ha avuto risposta entro il periodo di timeout.

ECONNREFUSED Connection refused. Un host remoto ha rifiutato la connessione (in genere dipende dal fatto che non c'è un server per soddisfare il servizio richiesto).

EHOSTDOWN Host is down. L'host remoto di una connessione è giù.

EHOSTUNREACH No route to host. L'host remoto di una connessione non è raggiungibile.

Interrogare il DNS

La funzione `gethostbyname` restituisce l'indirizzo IP (32 bit) dell'host di cui si conosce il nome.

```
#include <netdb.h>
struct hostent *gethostbyname(const char *name);
```

Ritorna un puntatore ad una struct `hostent`.

`gethostbyname()` non usa lo standard **`errno`**, ma il proprio **`h_errno`**, stampato da **`herror()`** invece che da **`perror()`**.

```
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 */
}

#define h_addr h_addr_list[0] /* for backward compatibility */
```

gethostbyname esempio

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <netdb.h>
#include <sys/types.h>
#include <netinet/in.h>

int main(int argc, char *argv[])
{
    struct hostent *h;
    if (argc != 2) { /* error check the command line */
        fprintf(stderr, "usage: getip address\n");
        exit(1);
    }

    if ((h=gethostbyname(argv[1])) == NULL)
    { /* get the host info */
        perror("gethostbyname");
        exit(1);
    }
    printf("Host name   : %s\n", h->h_name);
    printf("IP Address  : %s\n", inet_ntoa(*(struct in_addr *)h->h_addr));

    return 0;
}
```

Un po' di esercizio... continua

Goal:

- Inserire la read/write
- Controllare gli errori
- Accettare indirizzi testuali

Requisiti:

1. Lanciate il server [ELF] che trovate online sulla mia pagina web alla porta **8888** o altro.
2. Testate il client.

Tempo a disposizione:

60 minuti

Soluzione 1/3

```
/* liberamente ispirato al codice originale di dott. Vittorio Ghini */
```

```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

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

```
#define SOCKET_ERROR    ((int)-1)
#define SIZEBUF 10000
```

```
int main(int argc, char *argv[])
{
#define MAXSIZE 10000
    struct sockaddr_in Serv;
    struct hostent* h;
```

```
    char string_remote_ip_address[100];
    short int remote_port_number;
```

```
    int sockfd, ris;
    int n, i, nread, nwrite, len;
```

```
    char buf[MAXSIZE];
    char msg[MAXSIZE];
```

```
    for(i=0;i<MAXSIZE;i++) msg[i]='a';
    msg[MAXSIZE-1]='\0';
```

```
    if(argc<3)
    {
        printf("\n\nusage: %s IP port\n\n",argv[0]);
        exit(EXIT_FAILURE);
    }
```

Soluzione 2/3

```
/* set remote host */
strncpy(string_remote_ip_address, argv[1], 99);
remote_port_number = atoi(argv[2]);

/* convert URL to IP */
h=gethostbyname(string_remote_ip_address);
if(h==NULL)
{
    /* get the host info */
    perror("gethostbyname: ");
    exit(EXIT_FAILURE);
}
else
{
    /* get the host info */
    fprintf(stdout, "\nIP: %s\n", inet_ntoa(*(struct in_addr *)h->h_addr));
    fflush(stdout);
}

/* get a datagram socket */
socketfd = socket(AF_INET, SOCK_STREAM, 0);
if(socketfd== -1)
{
    perror("socket: ");
    exit(EXIT_FAILURE);
}

/* assign our destination address */
memset ( &Serv, 0, sizeof(Serv) );
Serv.sin_family = AF_INET;
Serv.sin_addr.s_addr = inet_addr(inet_ntoa(*(struct in_addr *)h->h_addr));
Serv.sin_port = htons(remote_port_number);

/* connection request */
ris = connect(socketfd, (struct sockaddr*) &Serv, sizeof(Serv));
if(ris== -1)
{
    perror("connect: ");
    exit(EXIT_FAILURE);
}
```

Soluzione 3/3

```
/* write */
len = strlen(msg)+1;
nwrite=0;
while( (n=write(socketfd, &(msg[nwrite]), len-nwrite)) >0 )
    nwrite+=n;

/* read */
nread=0;
while( (len>nread) && ((n=read(socketfd, &(buf[nread]), len-nread )) >0))
{
    nread+=n;
    printf("read effettuata, risultato n=%d  len=%d nread=%d len-nread=%d\n", n, len, nread, len-nread );
    fflush(stdout);
}

/* print result */
printf("stringa ricevuta: %s\n", buf);fflush(stdout);

/* close */
close(socketfd);

return(0);
}
```

Provate ...

```
./client www.csr.unibo.it 80
```

```
IP: 137.204.78.100
```

```
read effettuata, risultato n=486 len=10000 nread=486 len-nread=9514
```

```
stringa ricevuta: HTTP/1.1 414 Request-URI Too Large
```

```
Date: Fri, 14 Apr 2006 09:01:06 GMT
```

```
Server: Apache/1.3.34 (Debian) mod_ssl/2.8.25 OpenSSL/0.9.8a
```

```
PHP/4.3.10-16
```

```
Connection: close
```

```
Content-Type: text/html; charset=iso-8859-1
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
```

```
<HTML><HEAD>
```

```
<TITLE>414 Request-URI Too Large</TITLE>
```

```
</HEAD><BODY>
```

```
<H1>Request-URI Too Large</H1>
```

```
The requested URL's length exceeds the capacity
```

```
limit for this server.<P>
```

```
request failed: URI too long<P>
```

```
</BODY></HTML>
```

Esercizio su union

Goal:

- Utilizzare la union

Requisiti:

1. Memorizzate 2 numeri interi di 4 bytes scelti a caso 'a' e 'b'
2. Scambiate il byte n.2 tra 'a' e 'b'
3. Stampate a video il nuovo valore di 'a' e 'b'

Tempo a disposizione:

25 minuti

Soluzione union

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  //definisce il meta-tipo di variabile
5  typedef union
6  {
7      int value;
8      char p_value[4];
9  } byte_msg;
10
11
12 int main(int argc, char *argv[])
13 {
14     byte_msg a,b; //dichiara due variabili 'a' e 'b' del nuovo tipo 'byte_msg'
15     char tmp;
16
17     if(argc!=3)
18     {
19         printf ("\nNecessari 2 numeri\n");
20         exit(EXIT_FAILURE);
21     }
22
23     //setta il valore delle due variabili utilizzando la union come 'int'
24     a.value=atoi(argv[1]);
25     b.value=atoi(argv[2]);
26
27     //scambia i valori utilizzando la union come puntatore a 'char'
28     tmp=a.p_value[1];
29     a.p_value[1]=b.p_value[1];
30     b.p_value[1]=tmp;
31
32     //stampa i valori utilizzando la union come 'int'
33     printf("\na= %d ; b=%d",a.value,b.value);
34
35     return(0);
36 }
```

Soluzione dello studente G.Rossi

```
0 10 20 30 40 50 60 70 80
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 typedef union // viene definita una union di 4 bytes 'totali'
5 {
6     char num[4];
7     int val;
8 } set;
9
10 int main(int argc, char* argv[])
11 {
12
13     set num1, num2; // dichiara 2 union di tipo 'set'
14     char ex;        // char di appoggio per lo scambio di bytes
15
16     if (argc != 3)
17     {
18         printf("\n\nErrore: inserire 2 interi!!!\n\n");
19         exit(EXIT_FAILURE);
20     }
21
22     num1.val = atoi(argv[1]); // setta le 2 union come 'interi'
23     num2.val = atoi(argv[2]); // (impostandone tutti e 4 i bytes)
24
25     ex=num1.num[1];           // viene eseguito lo scambio del secondo byte
26     num1.num[1]=num2.num[1]; // (che e' il terzo piu' significativo
27     num2.num[1]=ex;           // di un intero in 'little endian')
28
29     // notare che l'effetto della modifica si fa sentire solo se uno dei 2 interi
30     // passati da riga di comando e' maggiore di 2^16 = 65536, cioe' se viene
31     // intaccato il terzo byte dell'intero (che e' il secondo della stringa)
32
33     printf("\n\nI 2 nuovi numeri sono:   %d   e   %d\n\n",num1.val, num2.val);
34
35     return 0;
36 }
```