

Domain Name Service

Mapping nomi/indirizzi con
Socket API in C

Risoluzione di nomi simbolici

- TCP/IP ha bisogno di rappresentazioni numeriche per gli indirizzi (es. 193.204.187.189) e per le porte (es. 80)
- Allora perché i nomi simbolici sono utili?
 - I nomi simbolici sono più facili da memorizzare
 - La rappresentazione numerica è efficiente per le macchine, non per l'uomo
 - L'indirizzo numerico lega una macchina alla rete a cui è connesso
 - Se non ci fossero i nomi simbolici, cosa succederebbe quando un server cambia provider e quindi indirizzo IP?

gethostbyname()

- Effettua la risoluzione da nome simbolico (es. www.uniba.it) a indirizzo Internet
- Signature
 - `struct hostent *gethostbyname(const char *hostname);`
- Restituisce NULL in caso di errore
- Funzione alternativa
 - `getaddrinfo()`

Struct hostent

```
struct hostent {  
    const char *h_name; /* canonical name of host */  
    char **h_aliases; /* alias list */  
    short h_addrtype; /* host address type (AF_INET) */  
    short h_length; /* length of address (INET)*/  
    char **h_addr_list; /* A NULL-terminated list of  
                        addresses (in_addr) for the  
                        host in network byte order.*/  
    #define h_addr h_addr_list[0];  
};
```

gethostbyaddr()

- Effettua il passaggio da indirizzo Internet a nome simbolico
 - Operazione inversa rispetto a gethostbyname()
- Signature

```
struct hostent * gethostbyaddr(const char*  
                                struct_in_addr,  
                                int addr_len_in_bytes,  
                                int addr_family_type);
```

- Nel nostro caso
 - addr_len_in_bytes è sempre 4
 - addr_family_type è sempre AF_INET
- Funzione alternativa
 - getnameinfo()

Risoluzione in locale: il file hosts

```
# Copyright (c) 1993-2006 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
# 102.54.94.97 rhino.acme.com      # source server
# 38.25.63.10 x.acme.com         # x client host

127.0.0.1    localhost
127.0.0.1    mio.laptop.it
```

Esempio

```
#include <stdio.h>
#include <stdlib.h>
#if defined WIN32
#include <winsock.h>
#else
#include <netdb.h>
#endif

int main(void) {
#if defined WIN32
WSADATA wsaData;
int iResult = WSStartup(MAKEWORD(2,2), &wsaData);
if (iResult != 0) {
    printf("Error at WSStartup()\n");
    return EXIT_FAILURE;
}
#endif
}
```

...Esempio...

```
const char * name = "localhost";
struct hostent *host;

host = gethostbyname(name);
if (host == NULL) {
    fprintf(stderr, "gethostbyname() failed.\n");
    exit(EXIT_FAILURE);
} else {
    struct in_addr* ina = (struct in_addr*) host->h_addr_list[0];
    printf("Risultato di gethostbyname(%s): %s\n", name,
          inet_ntoa(*ina));
}
```


...Esempio

```
const char* ip = "127.0.0.1";  
struct in_addr addr;  
addr.s_addr = inet_addr(ip);  
  
host = gethostbyaddr((char *) &addr, 4, AF_INET);  
char* canonical_name = host->h_name;  
fprintf(stdout, "Risultato di gethostbyaddr(%s): %s\n",  
                                                ip, canonical_name);  
return EXIT_SUCCESS;  
}
```

Altro Esempio

```
int main(int argc, char **argv) {
    ...
    struct hostent *remoteHost;
    char *host_name;
    struct in_addr addr;
    // Validate the parameters
    if (argc != 2) {
        printf("usage: %s ipv4 address\n", argv[0]);
        printf("  to return the host\n");
        printf("    %s 127.0.0.1\n", argv[0]);    return EXIT_FAILURE;
    }
    // Initialize Winsock in Windows
    ....
    host_name = argv[1];
    if (isalpha(host_name[0])) {    /* host address is a name e.g., "www.google.com" */
        printf("Calling gethostbyname with %s\n", host_name);
        remoteHost = gethostbyname(host_name);    }
    else    {    /* host address is IP address e.g., "127.0.0.1" */
        printf("Calling gethostbyaddr with %s\n", host_name);
        addr.s_addr = inet_addr(host_name);
        remoteHost = gethostbyaddr((char *) &addr, 4, AF_INET);
    }
    return EXIT_SUCCESS;}

```