

Laboratorio reti AA 2006/2007

Dott. Matteo Roffilli

roffilli@csr.unibo.it

**Ricevimento in ufficio
dopo la lezione**

Laboratorio reti AA 2006/2007

Per esercitarvi fate SSH su:

alfa.csr.unibo.it

si-tux00.csr.unibo.it

....

si-tux15.csr.unibo.it

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

Laboratorio reti AA 2006/2007

- **Marzo**
- 6 Intro,SSH,VI/VIM,GCC base
- 19 Richiami di C e Compilazione
- **26 Socket e Co.**

Socket su Windows

http://support.microsoft.com/kb/122928/it

Google Groups

Trova successivo

Modalità autore

Tutte le immagini

Adatta alla larghezza

50%

Italia

Quick Link | Home | Microsoft nel mondo

Microsoft

Cerca su Microsoft.com:

Vai

Supporto Tecnico Microsoft

Supporto Tecnico Microsoft | Seleziona un prodotto

Descrizione del WINSOCK.DLL File

[Visualizza i prodotti ai quali l'articolo e' applicato.](#)

NOTA: Questo articolo è stato tradotto da un sistema di traduzione automatica senza intervento umano. Microsoft mette a disposizione questi articoli come beneficio per coloro che non parlano la lingua inglese al fine di facilitarli nella comprensione. Microsoft non garantisce la qualità linguistica delle traduzioni e non è responsabile di qualsivoglia problema, diretto o indiretto, dovuto alla erronea interpretazione dei contenuti o dell'utilizzo degli stessi presso i clienti.

3,11 3,10
WINDOWS
Kb3rdparty kbnetwork

Identificativo articolo : 122928
Ultima modifica : giovedì 16 settembre 1999
Revisione : 1.0

Su questa pagina

- [Sommar](#)
- [Informazioni](#)
- [Cosa è WINSOCK.DLL?](#)
- [Dove è possibile ottenere WINSOCK.DLL?](#)
- [Dove è possibile ottenere Ulteriori informazioni a proposito di WINSOCK.DLL?](#)

Sommario

In questo articolo si illustra lo scopo del file WINSOCK.DLL e come ottenerlo.

[Torna all'inizio](#)

Informazioni

Cerca articoli (K)

KB italiana

[Ricerca avanzata](#)

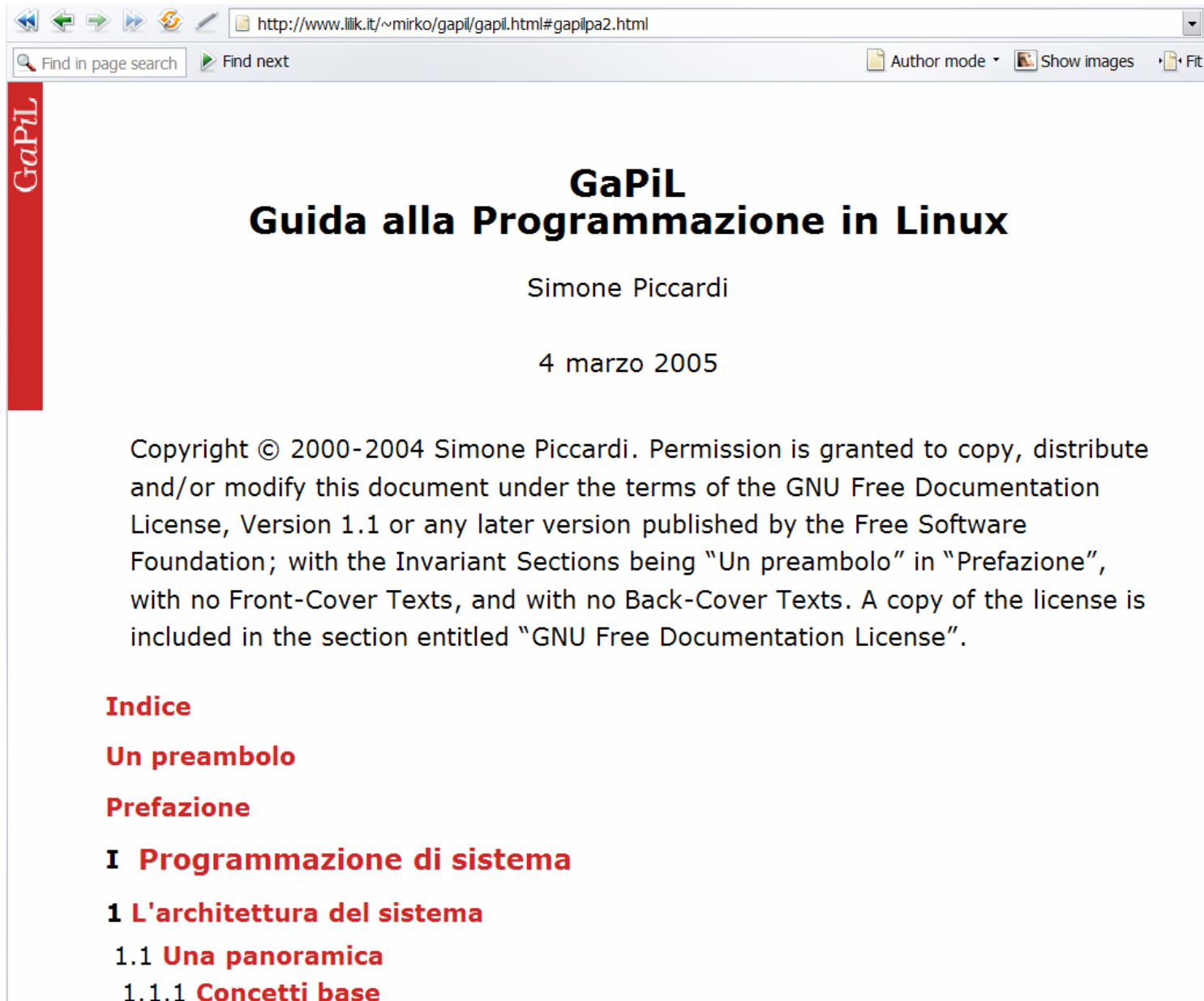
Traduzione articolo

Inglese (Stati Uniti)

Altre opzioni di supporto

- [Contattaci](#)
Numeri telefonici, o
supporto e prezzi, C
linea e altro ancora.
- [Servizio clienti](#)
Per assistenza non
relativa all'acquisto
sottoscrizioni, serv
eventi, corsi di form
vendite ad aziende,
problematiche legate
pirateria informatica
ancora.
- [Newsgroup](#)
Per inviare una dom
altri utenti. Gruppi d
discussione e forum
prodotti, tecnologie
specifici di Microsoft

Riferimento



http://www.liik.it/~mirko/gapil/gapil.html#gapilpa2.html

Find in page search Find next Author mode Show images Fit

GaPiL

GaPiL

Guida alla Programmazione in Linux

Simone Piccardi

4 marzo 2005

Copyright © 2000-2004 Simone Piccardi. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.1 or any later version published by the Free Software Foundation; with the Invariant Sections being "Un preambolo" in "Prefazione", with no Front-Cover Texts, and with no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Indice

Un preambolo

Prefazione

I Programmazione di sistema

1 L'architettura del sistema

1.1 Una panoramica

1.1.1 Concetti base

Socket linux

I socket sono uno dei principali meccanismi di comunicazione utilizzato in ambito Unix/Linux.

Un socket costituisce in sostanza un canale di comunicazione fra due processi.

I socket non sono limitati alla comunicazione fra processi che girano sulla stessa macchina, ma possono realizzare la comunicazione anche attraverso la rete.

Quella dei socket costituisce infatti la principale interfaccia usata nella programmazione di rete. La loro origine risale al 1983, quando furono introdotti in BSD 4.2; l'interfaccia è rimasta sostanzialmente la stessa, con piccole modifiche, negli anni successivi.

La flessibilità e la genericità dell'interfaccia inoltre consente di utilizzare i socket con i più disparati meccanismi di comunicazione, e non solo con l'insieme dei protocolli TCP/IP

Aprire un socket

```
#include <sys/socket.h>
```

```
int socket(int domain, int type, int protocol)
```

La funzione restituisce un intero positivo in caso di successo, e -1 in caso di fallimento, nel qual caso la variabile **errno** assumerà i valori:

- EPROTONOSUPPORT Il tipo di socket o il protocollo scelto non sono supportati nel dominio.
- ENFILE Il kernel non ha memoria sufficiente a creare una nuova struttura per il socket.
- EMFILE Si è ecceduta la tabella dei file.
- EACCES Non si hanno privilegi per creare un socket nel dominio o con il protocollo specificato.
- EINVAL Protocollo sconosciuto o dominio non disponibile.
- ENOBUFS Non c'è sufficiente memoria per creare il socket (può essere anche ENOMEM).

`int domain` specifica il dominio del socket (definisce cioè la famiglia di protocolli usata)

`int type` specifica il tipo di socket (definisce cioè lo stile di comunicazione)

`int protocol` in genere quest'ultimo è indicato implicitamente dal tipo di socket, per cui di norma questo valore viene messo a zero (con l'eccezione dei raw socket).

Si noti che la creazione del socket si limita ad allocare le opportune strutture nel kernel (sostanzialmente una voce nella file table) e non comporta nulla riguardo all'indicazione degli indirizzi remoti o locali attraverso i quali si vuole effettuare la comunicazione.

Il dominio, o protocol family

Dati i tanti e diversi protocolli di comunicazione disponibili, esistono vari tipi di socket, che vengono classificati raggruppandoli in quelli che si chiamano domini.

La scelta di un dominio equivale in sostanza alla scelta di una famiglia di protocolli, e viene effettuata attraverso l'argomento domain della funzione socket.

Ciascun dominio ha un suo nome simbolico che convenzionalmente inizia con una costante che inizia per **PF_**, iniziali di protocol family, un altro nome con cui si indicano i domini.

A ciascun tipo di dominio corrisponde un analogo nome simbolico, anch'esso associato ad una costante, che inizia invece per **AF_** (da address family) che identifica il formato degli indirizzi usati in quel dominio.

Uno dei più utilizzati è:

PF_INET AF_INET 2

Il tipo

La scelta di un dominio non comporta però la scelta dello stile di comunicazione, questo infatti dipende dal protocollo che si andrà ad utilizzare fra quelli disponibili nella famiglia scelta.

L'interfaccia dei socket permette di scegliere lo stile di comunicazione indicando il tipo di socket con l'argomento type di socket.

Linux mette a disposizione vari tipi di socket identificati dalle seguenti costanti:

1. **SOCK_STREAM** Provvede un canale di trasmissione dati bidirezionale, sequenziale e affidabile. Opera su una connessione con un altro socket. I dati vengono ricevuti e trasmessi come un flusso continuo di byte (da cui il nome stream).
2. **SOCK_DGRAM** Viene usato per trasmettere pacchetti di dati (datagram) di lunghezza massima prefissata, indirizzati singolarmente. Non esiste una connessione e la trasmissione è effettuata in maniera non affidabile.
3. **SOCK_SEQPACKET** Provvede un canale di trasmissione di dati bidirezionale, sequenziale e affidabile. Opera su una connessione con un altro socket. I dati possono vengono trasmessi per pacchetti di dimensione massima fissata, ed devono essere letti integralmente da ciascuna chiamata a read.
4. **SOCK_RAW** Provvede l'accesso a basso livello ai protocolli di rete e alle varie interfacce. I normali programmi di comunicazione non devono usarlo, è riservato all'uso di sistema.
5. **SOCK_RDM** Provvede un canale di trasmissione di dati affidabile, ma in cui non è garantito l'ordine di arrivo dei pacchetti.
6. **SOCK_PACKET** Obsoleto, non deve essere usato.

Indirizzi IPv4 sockaddr_in

I socket di tipo **AF_INET**/**PF_INET** vengono usati per la comunicazione attraverso internet; la struttura per gli indirizzi per un socket internet (se si usa IPv4) è definita come `sockaddr_in` nell'header file `netinet/in.h`

```
struct sockaddr_in
{
    sa_family_t    sin_family; /* address family: AF_INET */
    in_port_t      sin_port;   /* port in network byte order */
    struct in_addr sin_addr;    /* internet address */
};

/* Internet address. */
struct in_addr
{
    in_addr_t      s_addr;     /* address in network byte order */
};
```

L'indirizzo di un socket internet (secondo IPv4) comprende l'indirizzo internet di un'interfaccia più un numero di porta. Il protocollo IP non prevede numeri di porta, che sono utilizzati solo dai protocolli di livello superiore come TCP e UDP. Questa struttura però viene usata anche per i socket RAW che accedono direttamente al livello di IP, nel qual caso il numero della porta viene impostato al numero di protocollo.

struct sockaddr_in

sin_family deve essere sempre impostato a `AF_INET`, altrimenti si avrà un errore di `EINVAL`;

sin_port specifica il numero di porta. I numeri di porta sotto il 1024 sono chiamati riservati in quanto utilizzati da servizi standard;

sin_addr contiene un indirizzo internet, e viene acceduto sia come struttura che direttamente come intero.

netinet/in.h vengono definite anche alcune costanti che identificano alcuni indirizzi speciali.

endianess & Co

Il problema connesso all'endianess è che quando si passano dei dati da un tipo di architettura all'altra i dati vengono interpretati in maniera diversa, e ad esempio nel caso dell'intero a 16 bit ci si ritroverà con i due byte in cui è suddiviso scambiati di posto. Per questo motivo si usano delle funzioni di conversione che servono a tener conto automaticamente della possibile differenza fra l'ordinamento usato sul computer e quello che viene usato nelle trasmissioni sulla rete; queste funzioni sono `htonl`, `htons`, `ntohl` e `ntohs` ed i rispettivi prototipi sono:

```
#include <netinet/in.h>
```

```
unsigned long int htonl(unsigned long int hostlong)
```

Converte l'intero a 32 bit `hostlong` dal formato della macchina a quello della rete.

```
unsigned short int htons(unsigned short int hostshort)
```

Converte l'intero a 16 bit `hostshort` dal formato della macchina a quello della rete.

```
unsigned long int ntohl(unsigned long int netlong)
```

Converte l'intero a 32 bit `netlong` dal formato della rete a quello della macchina.

```
unsigned short int ntohs(unsigned short int netshort)
```

Converte l'intero a 16 bit `netshort` dal formato della rete a quello della macchina.

Tutte le funzioni restituiscono il valore convertito, e non prevedono errori.

Ipv4 → binario

Passare dal formato binario usato nelle strutture degli indirizzi alla rappresentazione simbolica dei numeri IP che si usa normalmente.

Le funzioni di manipolazione riguardano la conversione degli indirizzi IPv4 da una stringa in cui il numero di IP è espresso secondo la cosiddetta notazione dotted-decimal, (cioè nella forma 192.168.0.1) al formato binario (direttamente in network order) e viceversa.

Dette funzioni sono **inet_addr**, **inet_aton** e **inet_ntoa**, ed i rispettivi prototipi sono:

```
#include <arpa/inet.h>
```

```
in_addr_t inet_addr(const char *strptr)
```

Converte la stringa dell'indirizzo dotted decimal nel numero IP in network order.

```
int inet_aton(const char *src, struct in_addr *dest)
```

Converte la stringa dell'indirizzo dotted decimal in un indirizzo IP.

```
char *inet_ntoa(struct in_addr addrptr)
```

Converte un indirizzo IP in una stringa dotted decimal.

Tutte queste le funzioni non generano codice di errore.

Ipv4 → binario 2

La prima funzione, **inet_addr**, restituisce l'indirizzo a 32 bit in network order (del tipo `in_addr_t`) a partire dalla stringa passata nell'argomento `strptr`. In caso di errore (quando la stringa non esprime un indirizzo valido) restituisce invece il valore `INADDR_NONE` che tipicamente sono trentadue bit a uno. Questo però comporta che la stringa `255.255.255.255`, che pure è un indirizzo valido, non può essere usata con questa funzione; per questo motivo essa è generalmente deprecata in favore di `inet_aton`.

La funzione **inet_aton** converte la stringa puntata da `src` nell'indirizzo binario che viene memorizzato nell'opportuna struttura `in_addr` situata all'indirizzo dato dall'argomento `dest` (è espressa in questa forma in modo da poterla usare direttamente con il puntatore usato per passare la struttura degli indirizzi). La funzione restituisce 0 in caso di successo e 1 in caso di fallimento. Se usata con `dest` inizializzato a `NULL` effettua la validazione dell'indirizzo.

L'ultima funzione, **inet_ntoa**, converte il valore a 32 bit dell'indirizzo (espresso in network order) restituendo il puntatore alla stringa che contiene l'espressione in formato dotted decimal. Si deve tenere presente che la stringa risiede in memoria statica.