

# Laboratorio reti AA 2008/2009

**Dott. Matteo Roffilli**

**roffilli@csr.unibo.it**

**Ricevimento in ufficio  
dopo la lezione**

# Laboratorio reti AA 2008/2009

**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 2008/2009

- **Marzo**
- 5 Intro,SSH,VI/VIM,GCC base
- 12 Richiami di C e Compilazione
- **19 Socket e Co.**
- 26 Socket e Co. parte seconda (se lezioni non sospese per Lauree)

# Socket su Windows

The screenshot shows a web browser window displaying a Microsoft Support article. The address bar shows the URL <http://support.microsoft.com/kb/122928>. The page title is "Microsoft Aiuto & Supporto". The article is titled "Descrizione del file di WINSOCK.DLL" and has an article ID of 122928. It was last modified on August 9, 2007, and is revision 1.1. A warning icon indicates that the article is automatically translated. A disclaimer states that Microsoft is not responsible for the content of the article. The article is categorized under "3.10 3.11 WINDOWS kb3rdparty kbnetwork". There are buttons for "In questa pagina", "Espandi tutto | Chiudi tutto", "Sommario", and "Informazioni". The main content area explains the purpose of the WINSOCK.DLL file and provides information about its use in Windows applications. A sidebar on the right contains links for "Hai bisogno di aiuto?", "Traduzione articoli", and "Strumenti".

Identificativo articolo: 122928 - Ultima modifica: giovedì 9 agosto 2007 - Revisione: 1.1

## Descrizione del file di WINSOCK.DLL

⚠ Attenzione: Questo è un articolo tradotto in automatico.

📄 Dichiarazione di non responsabilità per articoli della Microsoft Knowledge Base su prodotti non più supportati

Visualizza i prodotti a cui si riferisce l'articolo.

3.10 3.11  
WINDOWS  
kb3rdparty kbnetwork

[In questa pagina](#)

[Espandi tutto](#) | [Chiudi tutto](#)

[Sommario](#)

In questo articolo viene illustrato lo scopo del file WINSOCK.DLL e come ottenerlo.

[Torna all'inizio](#)

[Informazioni](#)

### Che cos'è WINSOCK.DLL?

WINSOCK.DLL è una libreria di collegamento dinamico che fornisce un' comune interfaccia di programmazione delle applicazioni (API) per gli sviluppatori di applicazioni di rete che utilizzano il stack TCP / IP (Transmission Control Protocol/Internet Protocol).

Ciò significa che i programmatori che sviluppa un'applicazione TCP / IP basate su Windows, ad esempio un client FTP o Telenet possono scrivere un'applicazione che funziona con qualsiasi stack del protocollo TCP / IP che fornisce servizi di socket di Windows (WINSOCK.DLL).

Altre applicazioni che dipendono da un provider del socket di Windows includono (un

**Italia** Cambia | Tutti i siti Microsoft

Articoli in italiano | Articoli in inglese

Cerca Web

powered by Live Search

Home Page | Seleziona un prodotto | Ricerca avanzata

**Hai bisogno di aiuto?**  
Contatta un tecnico Microsoft.

**Traduzione articoli**  
Inglese (Stati Uniti) [→](#)

**Strumenti**  
♦ Stampa la pagina  
♦ Manda questa pagina a un collega

# Windows Communication Foundation

The screenshot shows a web browser window with the URL <http://msdn.microsoft.com/it-it/library/aa480210.aspx>. The browser's address bar and search bar are visible. The MSDN website header features the 'msdn' logo, a search bar with the text 'Avvia la ricerca in MSDN con Live Search', and links to 'MSDN Home' and 'Developer Center'. Below the header is a navigation bar with links to 'Home', 'Library', 'Formazione', 'Download', 'Supporto', and 'Community'. The main content area displays the breadcrumb trail: 'MSDN > MSDN Library > Articoli tecnici > Sviluppo .NET > Windows Communication Foundation > Informazioni generali sull'architettura...'. The article title is 'Informazioni generali sull'architettura Windows Communication Foundation'. The author is 'Microsoft Corporation' and the date is 'Marzo 2006'. The 'Riassunto' (Summary) states: 'L'articolo fornisce una panoramica generale dell'architettura Windows Communication Foundation (WCF) e dei suoi concetti di base. Mediante esempi di codice vengono illustrati contratti, endpoint e comportamenti di WCF (17 pagine stampate)'. The 'Sommario' (Summary) includes links to 'Introduzione', 'Concetti di base di WCF', 'Esempi di codice', and 'Riepilogo'. The 'Introduzione' section begins with: 'Nel presente documento viene fornita una panoramica generale dell'architettura Windows Communication Foundation (WCF) allo scopo di illustrarne i concetti di base e le relative interconnessioni. Per chiarire ulteriormente i concetti esposti, vengono presentati alcuni esempi di codice, che non rappresentano però la parte sostanziale del documento. Il documento è organizzato in due sezioni principali:'. A bulleted list follows: '• Concetti di base di WCF: prende in esame i principali concetti di WCF, i termini e i componenti dell'architettura.' and '• Esempi di codice: contiene alcuni brevi esempi di codice intesi a illustrare, anche nella loro applicazione concreta, i concetti esposti nella sezione precedente.' At the bottom left, there is a link '↑ Inizio pagina'.

msdn

Avvia la ricerca in MSDN con Live Search

MSDN Home Developer Center

MSDN

Home Library Formazione Download Supporto Community

Versione per la stampa Invia Valuta il contenuto e lascia un commento

MSDN > MSDN Library > Articoli tecnici > Sviluppo .NET > Windows Communication Foundation > Informazioni generali sull'architettura...

## Informazioni generali sull'architettura Windows Communication Foundation

Microsoft Corporation

Marzo 2006

**Riassunto:** L'articolo fornisce una panoramica generale dell'architettura Windows Communication Foundation (WCF) e dei suoi concetti di base. Mediante esempi di codice vengono illustrati contratti, endpoint e comportamenti di WCF (17 pagine stampate).

**Sommario**

- [Introduzione](#)
- [Concetti di base di WCF](#)
- [Esempi di codice](#)
- [Riepilogo](#)

### Introduzione

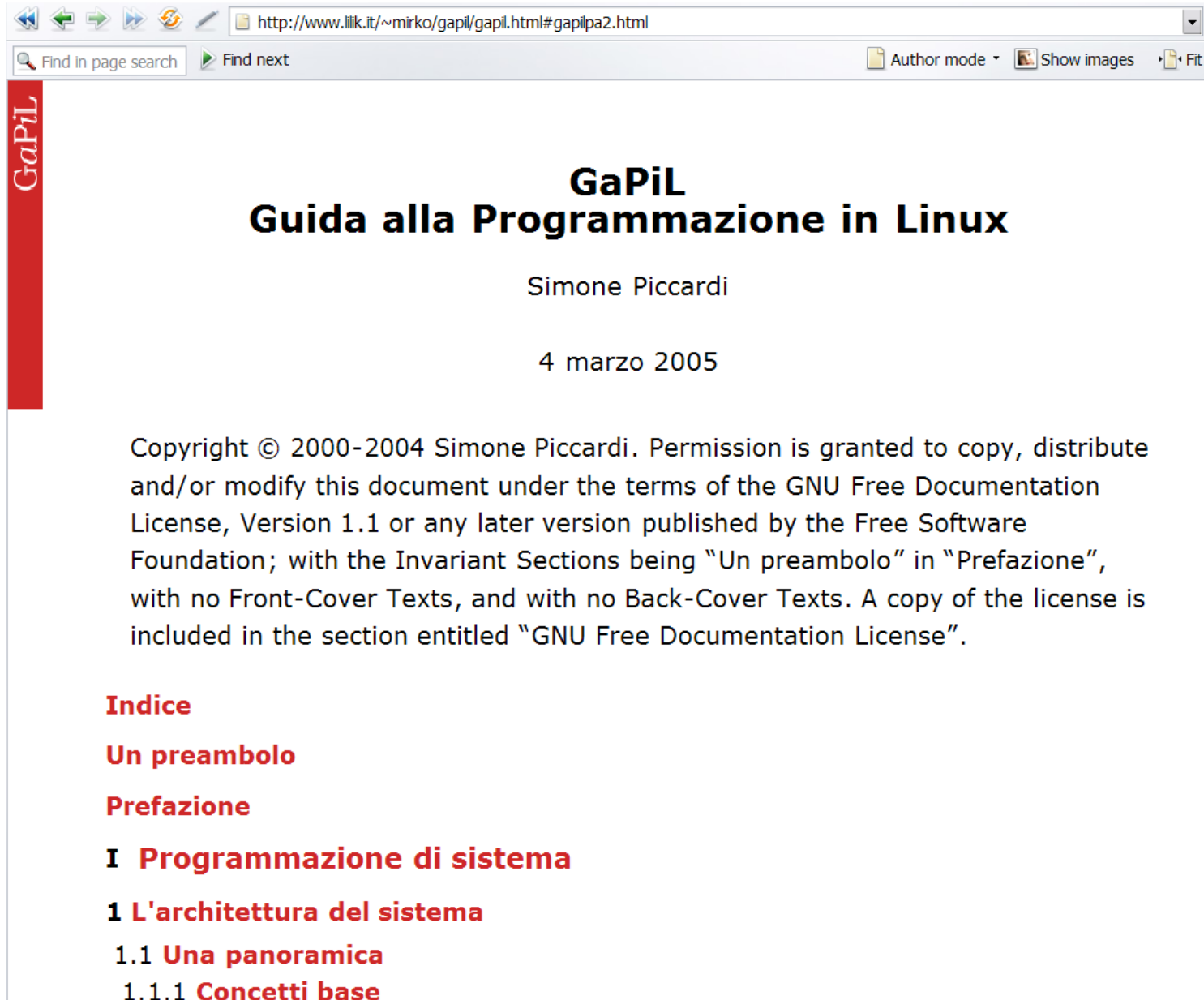
Nel presente documento viene fornita una panoramica generale dell'architettura Windows Communication Foundation (WCF) allo scopo di illustrarne i concetti di base e le relative interconnessioni. Per chiarire ulteriormente i concetti esposti, vengono presentati alcuni esempi di codice, che non rappresentano però la parte sostanziale del documento.

Il documento è organizzato in due sezioni principali:

- Concetti di base di WCF: prende in esame i principali concetti di WCF, i termini e i componenti dell'architettura.
- Esempi di codice: contiene alcuni brevi esempi di codice intesi a illustrare, anche nella loro applicazione concreta, i concetti esposti nella sezione precedente.

[↑ Inizio pagina](#)

# Riferimento



The image is a screenshot of a web browser window. The address bar shows the URL `http://www.liik.it/~mirko/gapil/gapil.html#gapilpa2.html`. Below the address bar is a search bar with the text "Find in page search" and a "Find next" button. To the right of the search bar are buttons for "Author mode", "Show images", and "Fit". On the left side of the browser window, there is a vertical red bar with the text "GaPiL" written vertically. The main content area of the browser displays the title "GaPiL Guida alla Programmazione in Linux" in a large, bold, black font. Below the title, the author's name "Simone Piccardi" is displayed in a smaller, black font. Below the author's name, the date "4 marzo 2005" is displayed in a smaller, black font. Below the date, there is a paragraph of text in a smaller, black font. Below the paragraph, there is a list of links in a smaller, red font. The links are: "Indice", "Un preambolo", "Prefazione", "I Programmazione di sistema", "1 L'architettura del sistema", "1.1 Una panoramica", and "1.1.1 Concetti base".

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**

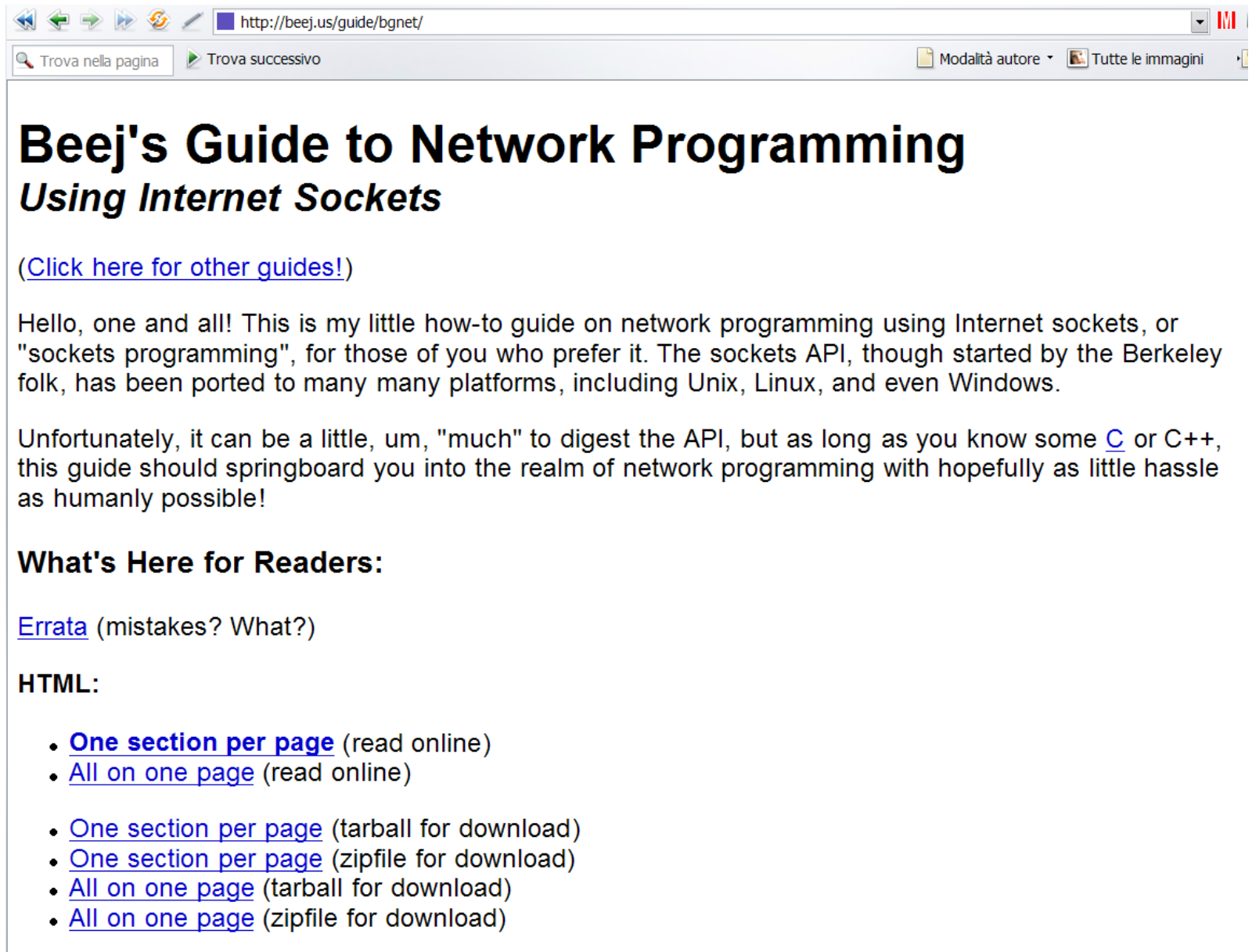
**I Programmazione di sistema**

**1 L'architettura del sistema**

**1.1 Una panoramica**

**1.1.1 Concetti base**

# Riferimento 2



The screenshot shows a web browser window with the address bar displaying `http://beej.us/guide/bgnet/`. The browser's navigation bar includes buttons for "Trova nella pagina" and "Trova successivo", along with a "Modalità autore" dropdown and a "Tutte le immagini" button. The main content area features the title "Beej's Guide to Network Programming" in a large, bold, black font, followed by the subtitle "Using Internet Sockets" in a slightly smaller, bold, black font. Below the title is a blue hyperlink "(Click here for other guides!)". The text of the guide begins with "Hello, one and all! This is my little how-to guide on network programming using Internet sockets, or 'sockets programming', for those of you who prefer it. The sockets API, though started by the Berkeley folk, has been ported to many many platforms, including Unix, Linux, and even Windows." This is followed by a paragraph stating that the guide should help digest the API, mentioning C and C++. A section titled "What's Here for Readers:" contains a blue hyperlink "Errata (mistakes? What?)". Below this, a section titled "HTML:" lists eight items in a bulleted format, each with a blue hyperlink and a description in parentheses: "One section per page (read online)", "All on one page (read online)", "One section per page (tarball for download)", "One section per page (zipfile for download)", "All on one page (tarball for download)", and "All on one page (zipfile for download)".

**Beej's Guide to Network Programming**  
***Using Internet Sockets***

([Click here for other guides!](#))

Hello, one and all! This is my little how-to guide on network programming using Internet sockets, or "sockets programming", for those of you who prefer it. The sockets API, though started by the Berkeley folk, has been ported to many many platforms, including Unix, Linux, and even Windows.

Unfortunately, it can be a little, um, "much" to digest the API, but as long as you know some [C](#) or C++, this guide should springboard you into the realm of network programming with hopefully as little hassle as humanly possible!

**What's Here for Readers:**

[Errata](#) (mistakes? What?)

**HTML:**

- [One section per page](#) (read online)
- [All on one page](#) (read online)
- [One section per page](#) (tarball for download)
- [One section per page](#) (zipfile for download)
- [All on one page](#) (tarball for download)
- [All on one page](#) (zipfile for download)

# 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.

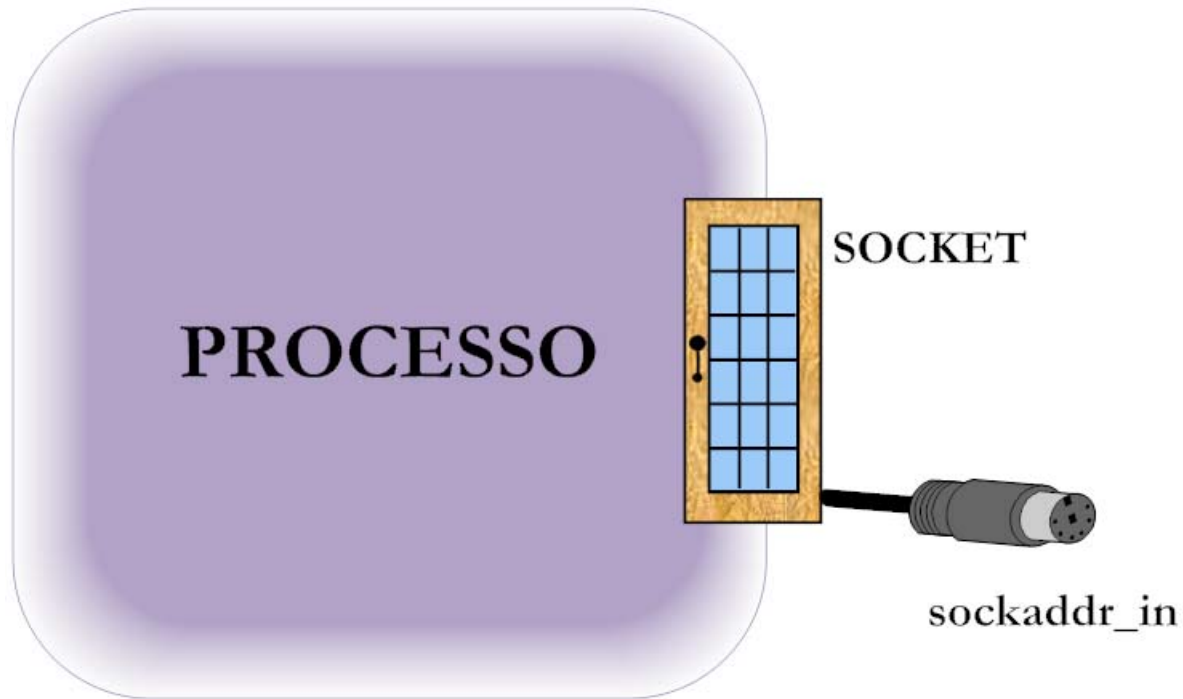
I socket non sono limitati alla comunicazione fra processi che girano sulla stessa macchina, ma possono realizzare la comunicazione anche attraverso la rete.

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



# Socket: una porta verso l'esterno



# Aprire un socket

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

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

`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).

# Aprire un socket 2

```
#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).

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.

# int domain (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.

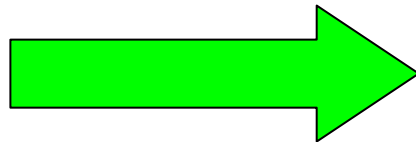
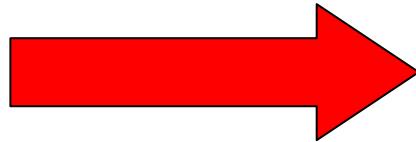
Ciascun dominio ha un suo nome simbolico che convenzionalmente inizia con una costante che inizia per **PF\_**, iniziali di protocol family, un altro nome con cui si indicano i domini.

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**

# Famiglie di protocolli definiti in Linux



Nome	Valore	Utilizzo	Man page
PF_UNSPEC	0	Non specificato	
PF_LOCAL	1	Local communication	unix(7)
PF_UNIX, PF_FILE	1		
PF_INET	2	IPv4 Internet protocols	ip(7)
PF_AX25	3	Amateur radio AX.25 protocol	
PF_IPX	4	IPX - Novell protocols	
PF_APPLETALK	5	Appletalk	ddp(7)
PF_NETROM	6	Amateur radio NetROM	
PF_BRIDGE	7	Multiprotocol bridge	
PF_ATMPVC	8	Access to raw ATM PVCs	
PF_X25	9	ITU-T X.25 / ISO-8208 protocol	x25(7)
PF_INET6	10	IPv6 Internet protocols	ipv6(7)
PF_ROSE	11	Amateur Radio X.25 PLP	
PF_DECnet	12	Reserved for DECnet project	
PF_NETBEUI	13	Reserved for 802.2LLC project	
PF_SECURITY	14	Security callback pseudo AF	
PF_KEY	15	PF_KEY key management API	
PF_NETLINK	16	Kernel user interface device	netlink(7)
PF_PACKET	17	Low level packet interface	packet(7)
PF_ASH	18	Ash	
PF_ECONET	19	Acorn Econet	
PF_ATMSVC	20	ATM SVCs	
PF_SNA	22	Linux SNA Project	
PF_IRDA	23	IRDA sockets	
PF_PPPOX	24	PPPoX sockets	
PF_WANPIPE	25	Wanpipe API sockets	
PF_BLUETOOTH	31	Bluetooth sockets	

# int type

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`.

```
#include <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 (IP) */
};

/* 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.

# Esercizio sui tipi di dato!

## Goal:

realizzare un programma che legge a riga di comando il numero di elementi da inserire, richiede gli elementi (numeri di tipo **float** e **long**) e stampa a video la loro somma

## Requisiti:

1. Lo spazio per memorizzare i numeri richiesti deve essere allocato dinamicamente.
2. L'allocazione di memoria deve essere controllata.
3. L'accesso ai numeri deve utilizzare l'aritmetica dei puntatori
4. Il programma deve essere compilato, linkato e deve funzionare!!!

## Tempo a disposizione:

25 minuti

---- esempio ----

Somma.exe 3

127

27.7

7

*La somma è 161.7*