**Materiale su POSIX sockets API**

*Queste note si aggiungono, ma non si sostituiscono, a quanto scritto nei commenti al codice, in altri file (inclusi i README), nelle dispense e nel libro.*

L’API che stiamo usando per la programmazione delle socket in C fanno parte della specifica POSIX[[1]](#footnote-1). Questa API viene anche detta Berkeley sockets o BSD sockets, in omaggio alla sua prima implementazione nel sistema operativo BSD (Berkeley Software Distribution).

Il sistema operativo BSD, basato su Research UNIX, non è più sviluppato, ma oggi il termine viene comunemente associato a vari discendenti open source.

**Riferimenti utili:**

* Stevens, W.R., Fenner, B., & Rudoff, A.M. (2003). UNIX Network Programming: The Sockets Networking API. (3rd ed., Vol. 1). Addison Wesley
* (non verificato nel dettaglio) Beej's Guide to Network Programming: <https://beej.us/guide/bgnet/>
* <https://en.wikipedia.org/wiki/Berkeley_sockets>
* man pages, per esempio (tra parentesi il link alla versione web per Ubuntu 22.04 LTS:
  + man udp (<https://manpages.ubuntu.com/manpages/jammy/en/man7/udp.7.html>)
  + man tcp (<https://manpages.ubuntu.com/manpages/jammy/en/man7/tcp.7.html>)
  + man socket (<https://manpages.ubuntu.com/manpages/jammy/en/man7/socket.7.html>)
  + man setsockopt (<https://manpages.ubuntu.com/manpages/jammy/en/man2/setsockopt.2.html>)
  + man bind (<https://manpages.ubuntu.com/manpages/jammy/en/man2/bind.2.html>)
  + man listen (<https://manpages.ubuntu.com/manpages/jammy/man2/listen.2.html>)
  + man accept (<https://manpages.ubuntu.com/manpages/jammy/man2/accept.2.html>)
  + man connect (<https://manpages.ubuntu.com/manpages/jammy/en/man2/connect.2.html>)
  + man send (<https://manpages.ubuntu.com/manpages/jammy/man2/send.2.html>)
  + man recv (<https://manpages.ubuntu.com/manpages/jammy/man2/recv.2.html>)
  + man inet\_ntop (<https://manpages.ubuntu.com/manpages/jammy/man3/inet_ntop.3posix.html>)
  + man htonl (<https://manpages.ubuntu.com/manpages/jammy/man3/byteorder.3.html>)
  + man close (<https://manpages.ubuntu.com/manpages/jammy/man2/close.2.html>)
  + man 2 shutdown (<https://manpages.ubuntu.com/manpages/jammy/man2/shutdown.2.html>)
  + man procfs (<https://manpages.ubuntu.com/manpages/jammy/man5/proc.5.html>)  
    per informazioni sul procfs, uno pseudo-filesystem che fornisce una interfaccia per le strutture dati del kernel

**Modalità di funzionamento:**

L’API per le socket può operare in due modalità:

* bloccante
* non bloccante

Noi stiamo usando l’API in modalità bloccante, che è la modalità predefinita.

In generale, invocate in modalità bloccante, le *system call* per le socket ritornano solo dopo il completamento dell’operazione, il verificarsi di un errore oppure l’interruzione della system call per la cattura di un segnale (il classico valore di ritorno *EINTR*).

Alcuni esempi:

* *connect* (per le socket TCP) si blocca fino al completamento del *three-way handshake* e la transizione della socket nello stato ESTABLISHED
* *accept* si blocca finché non c’è una connessione da accettare (e quindi restituire);
* *recv* (e simili) si blocca finché il buffer di ricezione è vuoto. Si noti che in UDP leggiamo datagrammi interi (quindi dobbiamo passare un buffer sufficientemente grande, pena perdita dei dati). In TCP, al contrario, possiamo leggere quanti byte desideriamo. In generale, il numero di byte letto può essere inferiore a quanto richiesto. Un valore restituito uguale a 0, in UDP indica la ricezione di un datagramma con payload vuoto, mentre in TCP indica un EOF.
* *send* (e simili) si blocca finché il buffer di trasmissione è pieno. Si noti che la chiamata potrebbe indicare (nel valore di ritorno) l’invio di un numero inferiore di byte. Inoltre, non è ancora detto che i dati siano stati ricevuti dal destinatario.

1. <https://pubs.opengroup.org/onlinepubs/9699919799/functions/V2_chap02.html#tag_15_10> [↑](#footnote-ref-1)