SOCKET

Richiami di rete

//da completare

Sockets

Abilitano la comunicazione tra processi server e client. Possono essere orientati alla connessione o meno. Possono essere considerati endpoint in comunicazione.

Una socket è individuata dalla concatenazione di una porta (TCP layer) e un indirizzo IP (Network layer), quest'ultimo identifica un sistema host. Le stream sockets utilizzano TCP protocol e offrono un servizio affidabile di spedizione, le datagram sockets usano UDP protocol e la spedizione non è garantita, le raw sockets consentono l'accesso diretto ai protocolli di livello inferiore.

E' come detto un endopoint per una connessione basata su TCP/IP, il livello d'applicazione ci si collega, il programmatore si occupa delle API.

Molte porte sono riservate come quelle dalla 0 alla 1023 compresa, l'OS assegna porte a client connessi con poca durata di vita dalla 1024 alla 5000 compresa.

La socket nasconde tutto ciò che l'OS fa nei layers sottostanti all'applciativo, di ritorno avremo un file descriptor di riferimento alla socket.

La Socket Address Structure è una struttura in C di tipo **sockaddr_in** in cui all'interno ci sono 5 proprietà fondamentali: **sin_len** che indica la lunghezza della struttura, **sin_family** che bisogna impostare ad **AF_INET** affinchè si possa utilizzare un IPV4, la **sin_port** numero di porta, un array **sin_zero** non utilizzato, e una struttura di tipo **in_addr** di nome **sin_addr** che contiene al suo interno l'indirizzo IP a 32 bit.

Le primitive per una connessione TCP per le sockets sono: **socket()** che crea un nuovo endpoint di comunicazione, **bind()** che associa alla socket un indirizzo locale, **listen()** che data una coda di client possibili in attesa di accettazione annuncia di essere pronta ad accettare connessioni, **accept()** che stabilisce passivamente una connessione, **connect()** che tenta di stabilire attivamente una connessione, **send()** invia dati, **receive()** riceve dati, **close()** rilascia la connessione.

Da un lato avremo una socket remota (server) che prima indicherà di essere pronta ad accettare e poi accetterà connessioni, mentre il client semplicemente proverà a connettersi attivamente alla socket.

Socket(): primo parametro è un intero che indica la famiglia dell'indirizzo (AF_INET = IPV4, AF_INET6 = IPV6, AF_LOCAL = local Unix), il secondo parametro è un intero che indica il tipo di socket (SOCK_STREAM, SOCK_DGRAM, SOCK_RAW), e un terzo intero che indica qualche protocollo usato nelle raw socket quindi lo piazziamo a 0. Ritorna -1 se fallisce senno l'fd della socket.

Bind(): prende un intero che è l'fd della socket, prende un puntatore a struct sockaddr (da castare) costruita in precedenza la lunghezza della struttura. Ritorna 0 se tutto apposto o -1, se errore è EADDRINUSE significa che l'indirizzo è gia usato.

Listen(): prende l'fd della socket e un intero che indica quanti client possono rimanere in attesa.

Accept(): prende l'fd della socket, , prende un puntatore a struct sockaddr (da castare) dove il client inserirà le informazioni del protocollo da lui usato e la lunghezza della struct. Ritorna un descritptor di riferimento al client o -1 se c'è stato un errore, avvia la connessione con il client.

Close(): prende solo l'fd della socket e la chiude. -1 in errore.

Connect(): prende l'fd della socket, , prende un puntatore a struct sockaddr (da castare) dove inserisce i dati del server e la sua lunghezza. Si usa nel client che non ha bisogno della bind perché è l'OS ad assegnarli la porta, ritorna l'fd della socket se va tutto bene sennò -1.

Recv(): prende fd socket, puntatore a buffer da riempire, la sua grandezza e delle flag che non ci interessano.

Send(): prende fd socket, puntatore a buffer da copiare, quanto è pieno, e delle flag che non ci interessano.

NB la maggior parte dei protocolli di IP e TCP usano la rappresentazione BIG ENDIAN dei degli indirizzi e delle porte. Esistono primitive che permettono la conversione di questi dati!

Htons() che converte un numero di porta dalla rappresentazione locale a quella network

Ntohs che converte un numero di porta dalla rappresentazione network a quella locale

inet_addr() che converte un indirizzo dotted in uno network da inserire nella struttura

*inet_ntop() che converte un indirizzo network in dotted e lo salva in un buffer.