Come Scrivere Programmi di RETE

PIPE ⇒ Modo per far dialogare 2 Processi

Ormai la Maggior parte dei PROGRAMMI da STAND ALONE Sono diventati di RETE. (Word come WEB-APPLICATION)

GDB per il C sta a WIRESHARK per i Programmi di Rete

SOCKET ⇒ È l'ASSEMBLY DELLA PROGRAMMAZIONE di Rete

## INTERFACCIA SOCKET:

È una System Call del Sistema Operativo e parente della PIPE (Sulla Stessa Macchina)

Con RETE posso Comunicare Su PC diversi

INTERFACCIA = Insieme di Chiamate a funzione C Se messe a disposizione da S.O. vengono chiamate SYSTEM CALL.

S.O. lo fa per ISOLARE HW

Livello Trasporto è LVL 4 sopra ad IP

- TCP o UDP
  - Molto SIMILE alla PIPE del S.O. (che è un sistema STREAM)

Essendo ORIENTATO alla Connessione il Modo è MOLTO Similare.

HOST è una MACCHINA Sempre identificata tramite IP ed opzionalmente da un NOME INTERNET (www.dominio)

L'host OSPITA i programmi UTENTE (quelli che Scriviamo NOI)

ROUTER NON Sono HOST, infatti ha più di un IP Associato

Dominio è un pezzo di un NOME INTERNET

APPLICAZIONE ⇒ Collaborazione di un INSIEME di PROCESSI per fare Quello che dice l'Applicazione

Programma In ESECUZIONE

Un Applicazione Normale è Solamente 1 processo, appl. di RETE Sono Almeno 2 processi Che Comunicano Tra Loro

PARADIGMA Client-Server
Un Processo Altro Processo

Chi ha a che fare con il CLIENT è il FRONT-END, con Interfaccia Grafica e facilmente Utilizzabile

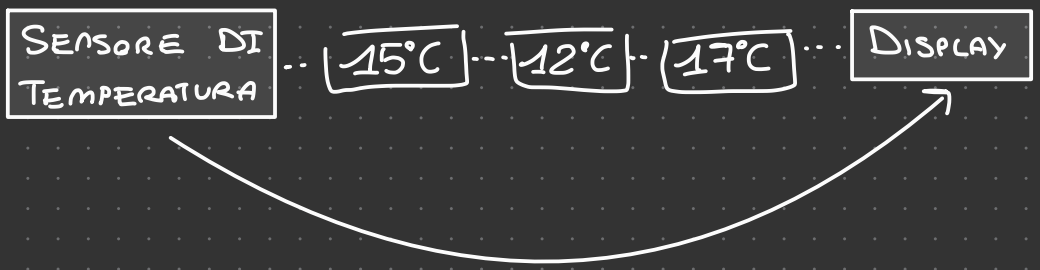
BROWSER è un Processo in Esecuzione, una parte dell' Applicazione, che è il FRONT-END, ma APACHE NGIX è il PROCESSO BACK-END che mi Risponde Alle mie Richieste. (Se non lo Avessi Allora Sarebbe INUTILE Visto che Vedrei Solamente una finestra Vuota)

Per NON Avere un SERVER in Mezzo che Smista i NOSTRI messaggi TELEGRAM ad Esempio dovremmo Avere 2 IP pubblici ma invece Solitamente Siamo dietro a dei NAT con IP privato

È fondamentale Quindi Sapere il NOSTRO IP

## ORIENTATO AL DATAGRAM: UDP

Ogni Pacchetto è Indipendente, perdite dei Pacchetti NON sono Tenute in considerazione:




## ORIENTATO ALLA CONNESSIONE TCP

Se Trasmetto Qualcosa di più COMPLESSO.

Tra pacchetti c'è una RELAZIONE, fanno parte di un Messaggio più GRANDE

Serve un NUMERO DI SEQUENZA per farsi REINVIARE Eventuali pacchetti PERSI.

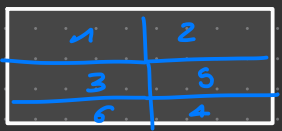
VANTAGGI:

- C'è la Stessa Naturalezza di Quando Scrivo/leggo su un Archivio locale Come Se la RETE non ci fosse

## SVANTAGGI:

- Trasmettitore e Ricevitore devono Lavorare da più "DENTRO" al Sistema Operativo
- Può Esserci più Ritardo (per RITRASMISSIONE)

IMG:




Suddivisione in PACCHETTI diversi, a destinazione li VOGLIO tutti

Ci Sono 2 SCHEMI DI APPLICAZIONI:

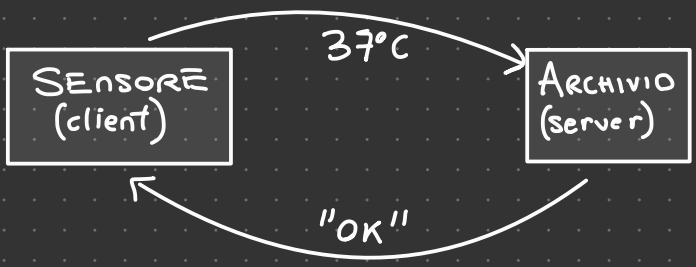
### CLIENT/SERVER:

Non si Tratta di TCP o UDP, posso farlo con ENTRAMBI, ma del come Interagiscono i 2 Attori.

1. Client fa SEMPRE il Primo Passo con una RICHIESTA
2. Server RISPONDE alla richiesta

Ovvio che il SERVER non risponderà Solamente ad 1 client

CLIENT è Colui che fa il primo Passo e NON Quello di fare una richiesta di un DATO o chissà che cose infatti Qui:



In Questo Caso il Client ci fornisce il DATO.

Client e Server Sono dei Processi e NON host

---

Utilizzeremo una libreria per Astrarre Molti Concetti

Non dovrò PREOCCUPARMI di Come ci Arriverà a destinazione, ci penserà lo STACK ISO/OSI

Host Identifico con IP ma per IDENTIFICARE il Processo mi Serve anche la PORTA oltre all' IP.

172.0.0.1 ⇒ Host
172.0.0.1:8080 ⇒ Processo X sul determinato Host

UDPRecive è Bloccante, attendo Qualcosa da leggere

Nel client:

SEND
⋮
RECIVE
⋮

Nel Server

RECIVE
⋮
SEND

Che Equivale ad una Schema di Sincronizzazione

Per Compilare ricordarsi -lpthread

---

Mantenere TUTTO IN UNA CARTELLA ORDINATA PER BENE

ESAME ORALE+PRATICO CON DAVANTI IL PROPRIO PC

SOCKET è fornito dal S.O. Quindi c'è in OGNI Linguaggio di Programmazione come:

- Java
- Python
- .....

} Possiamo Sceglierlo NOI

PUBLISHER/SUBSCRIBER: