Alessio Riccomi 4CI

Progetto discord.

Per questo progetto ho scelto di realizzare una chat discord

ANALISI DEL PROBLEMA

Scelta progettuale

In questo progetto esiste una comunicazione tra un client e un server

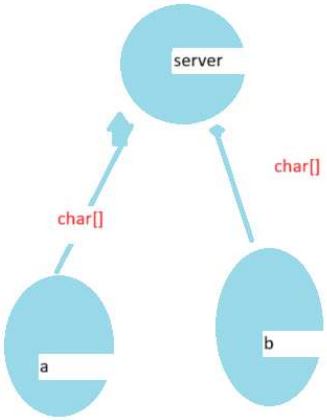
Per prima cosa occorre capire come voler progettare l'architettura.

Abbiamo due canali, canale A e canale B che possono comunicare tra di loro rispettivamente passando da un server centrale.

Per semplificarmi il lavoro ho deciso di eseguire 2 codici, uno per il client ed un altro per il server, anche se i due sono molto simili

SCELTA PROGETTUALE

Per prima cosa ho deciso di dare un'occhiata a ciò che poteva al meglio rappresentare un'architettura giusta

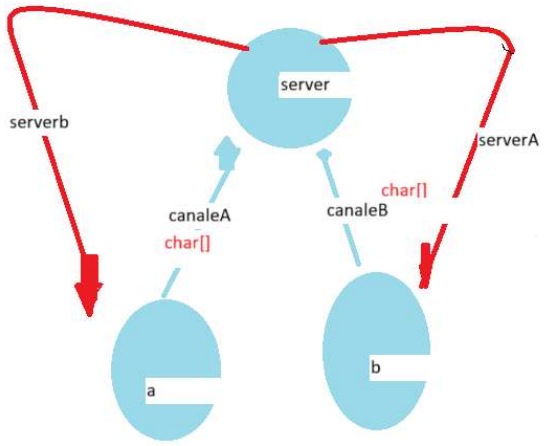


LIMITAZIONI DELL'ARCHITETTURA(MOTIVAZIONI)

Dopo un'ulteriore riflessioni, ho notato una grossa limitazione della seguente architettura.

Questa architettura infatti non permetteva al server di comunicare con i due canali, cosa fondamentale se vogliamo che i nostri messaggi arrivino a destinazione

L'architetturà dunque finale, è la seguente



Vantaggi: comunicazione bidirezionale

Svantaggi:più pipe,dunque bisogna fare più attenzione

spiegazione architettura

Sappiamo che A&B non possono comunicare DIRETTAMENTE tra loro, ma hanno bisogno di un server, quindi la comunicazione deve avvenire anche da server a canale, in modo che il server possa inoltrate il messaggio al canale destinatario

Come possiamo vedere le pipe sono 4

CanaleA: A-------->SERVER

CanaleB: B------->SERVER

ServerB:server-------->b

ServerA:server-------->A

Le pipe verranno create tramite la istruzione mkfifo

Da cosa è dovuto l'uso delle named pipe? Perchè permettono di non doversi aspettare perforza un valore all'ascolto, di conseguenza, potrò aspettare un messaggio in arrivo mentre sto scrivendo al server, senza bloccare l'esecuzione del codice

IDEE REALIZZATIVE (client):

Nel client canale A e canale B sono aperti sia in scrittura che in lettura

L'idea realizzativa sarebbe quella di far scegliere all'utente a chi voler mandare il messaggio, e su questa base si baserebbe il progetto

Esempio: se il mio utente sceglie di mandare un messaggio al canale B, iop utilizzerò la write, mandando il messaggio sulla pipe che mi collega il canaleA con il server,

Nel frattempo farò una fork, in questo modo il processo figlio, eseguirà in background un loop infinito di controllo, controllerà se dovessero arrivare messaggi dal canale b, quindi controllerà il canale server----------> canale A, come imposto dalle istruzioni.

Pseudocodice:

Scegli un canale

Se canale a

    Scrivi su server

Nel mentre

Figlio aspetta un messaggio

Altrimenti

Fai lo stesso con canale b

IDEE REALIZZATIVE (server)

Il server ha il compito di smistare i messaggi

Le pipe sono uguali a quelle di client, ma cambia la loro modalità

CanaleA: A-------->SERVER(solo lettura)

CanaleB: B------->SERVER(solo lettura)

ServerB:server-------->b(RDRW)

ServerA:server-------->A(RDRW)

Spiegazione

Il messaggio è già in circolazione, e quindi al server basterà leggere dalla pipe che collega canale a con server, una volta arrivata al server, il server la scriverà nella pipe che collega il server al canale b

Pseudocodice:

Ripeti sempre:

Se c'è un messaggio nella pipe canale a---->server

Inoltralo alla pipe server---->canaleb

Altrimenti

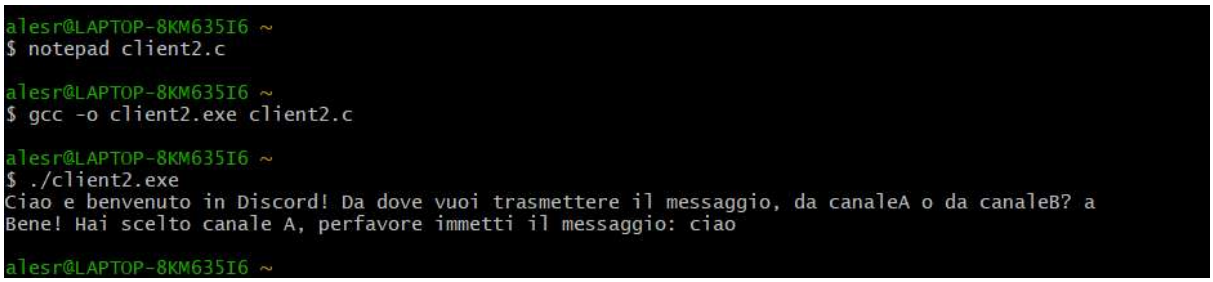
Controlli la pipe canaleb----> server

Se c'è un messaggio inoltri a canale a

Altrimenti

Ripeti il loop

Screenshot su come eseguire il sistema



Questi sono i messaggi con le istruzioni da eseguire per far funzionare il programma sul terminale cygwin, questo è il programma client, possiamo vedere anche un test del programma, dove scegliamo il canale, e scegliamo il messaggio

server

Il server continua a controllare se ci sono dei messaggi per inviarli alll'altra chat, in questo caso non vediamonulla perchè l'input deve arrivare da client