

BlockExplorer

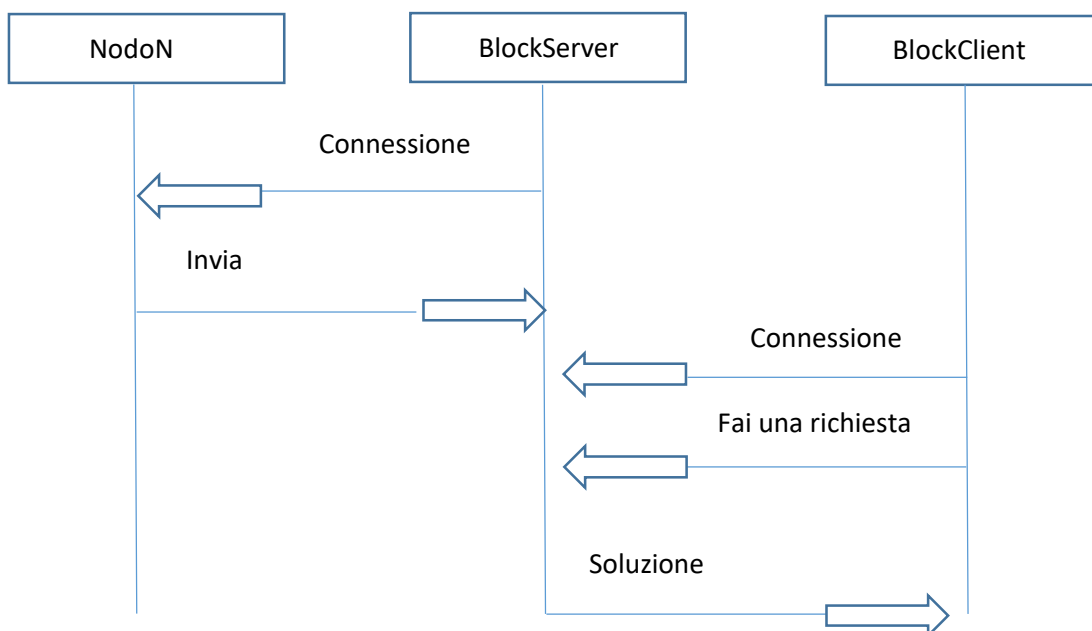
DESCRIZIONE PROGETTO:

Il progetto consiste nel creare un NodoN ossia un server che genererà transazioni, la transazione è costituita da una chiave ossia il numero progressivo dei blocchi, dall'indirizzo ip del mittente, dai soldi e dall'indirizzo ip del destinatario a cui è arrivata questa transazione.

Il BlockServer sarà sia un client che un server. Sarà un client in quanto lui si connetterà al NodoN per ricevere le transazioni e sarà anche Server in quanto aspetta la connessione da parte di un client per gestire le sue richieste.

Il client possono essere molteplici in quanto il BlockServer è concorrente e il client una volta connesso al BlockServer potrà richiedere una transazione specifica, il totale dei soldi di tutte le transazioni, ricercare una transazione specifica a partire dall'indirizzo ip del destinatario o mittente e trovare le ultime N transazioni generate. L'idea è quella che il client non deve sapere nulla di come vengono gestite le transazioni.

Dunque avremo una situazione di questo tipo:



DESCRIZIONE IMPLEMENTATIVA:

Abbiamo una lista allocata dinamicamente con inserimento in testa che si trova in lista.h mentre nel wrapper abbiamo tutte le funzioni.

Il NodoN con una select attende la connessione da parte del BlockServer ,nel momento della connessione inizia ad inviare al BlockServer i nodi in tempo reale.

Il BlockServer avrà una duplice struttura una di tipo client e una server.

Il blockserver è gestito in questo modo viene effettuata la prima fork() dove il padre e figlio avranno dei ruoli diversi.

Il figlio sarà la parte client rispetto al NodoN dunque si occupa di ricevere e allocare le transazioni mentre il padre si comporta da Server e sarà in attesa di una connessione.

Nel momento in cui arriva la connessione da parte del BlockClient ci sarà una nuova fork dove sarà gestito il client e una volta esaudita la sua richiesta verrà chiuso.

Fare chiarezza su alcune parti implementative:

1) Nel BlockServer vengono usate la pipe. In quanto abbiamo detto che praticamente il figlio si comporta da Client rispetto al NodoN mentre il padre attende la connessione da parte del BlockClient. Poichè sono aree di memoria diverse ,le transazioni che vengono salvate nella memoria del figlio in real time non verranno anch'esse visualizzate dal padre e dunque perderemo la lista.

Dunque per fare comunicare questi due processi ho usato la pipe in quando che anche il padre possa avere la lista nella sua memoria locale.

2) Ho gestito la caduta del NodoN.

Poichè il figlio legge in real-time quello che gli viene inviato dal NodoN, la caduta di quest'ultimo provoca un ciclo infinito in quanto il figlio leggerebbe il nulla. Dunque si è certi di una cosa che le transazioni non posso avere soldi=0 dunque se il figlio legge che una transazione ha come campo della lista soldi=0 faremo un

raise(SIGUSR1) ossia si suicida in modo da permettere comunque al padre di continuare nell'aspettare le connessioni.

3) Il padre praticamente aspetta la connessione da parte del BlockClient però nel frattempo che fa polling legge i dati che gli sta passando il figlio nella pipe .

BLOCKSERVER

```
figlio = fork();

if (figlio == 0) //sono il figlio
{
    for (;;) {
        close(socket);
        close(fd[0]);
        new_nodo = malloc(sizeof(NodoN));
        FullRead(listenfd, new_nodo, sizeof(NodoN));

        if (new_nodo -> soldi == 0) raise(SIGUSR1);

        FullWrite(fd[1], new_nodo, sizeof(NodoN));
        new_nodo -> next = head;
        head = new_nodo;

        printf("ricevuto\n");
    }
}
if (figlio != 0) { //sono il padre

    NodoN * testa = NULL;
    NodoN * nuovo;
    for (;;) {
        nuovo_socket = accept(socket, NULL, NULL);
        if (nuovo_socket == -1) {
            if (errno == EWOULDBLOCK) //Praticamente sarà in uno stato di polling fin quando non ci sarà qualcuno a connettersi.
            {
                nuovo = malloc(sizeof(NodoN));
                FullRead(fd[0], nuovo, sizeof(NodoN));
                nuovo -> next = testa;
                testa = nuovo;
            }
        }
    }
}
```

Qui vengono soddisfatte le richieste da parte del client

```

} else { //in questa parte il nipote gestirà il client connesso.
    nipote = fork();
    if (nipote == 0) {
        close(socket);

        printf("connesso\n");
        FullRead(nuovo_socket, & scelta, sizeof(scelta));

        switch (scelta) {
            case 1:
                printf("E' stata selezionata la scelta 1\n"); /*qui calcola i soldi della transazioni*/
                NodoN * ok = testa;

                int totale = 0;
                while (ok != NULL) {
                    totale = totale + ok -> soldi;
                    ok = ok -> next;
                }
                FullWrite(nuovo_socket, & totale, sizeof(int));

                close(nuovo_socket);
                break;

            case 2:
                printf("E' stata selezionata la scelta 2\n"); /*va a cercare una transazione specifica*/

                FullRead(nuovo_socket, & numero_transazione, sizeof(int));

                NodoN * ok1 = testa;

                while (ok1 != NULL) {

                    if (ok1 -> numero_blocco == numero_transazione) {
                        FullWrite(nuovo_socket, ok1, sizeof(NodoN));
                        break;
                    }
                    ok1 = ok1 -> next;
                }
                close(nuovo_socket);
                break;
        }
    }
}

```

Praticamente poiché abbiamo un server concorrente il padre ogni volta che viene richiesta la connessione da parte di un BlockClient genera un nipote, il nipote andrà a soddisfare le richieste del BlockClient.

Nel primo caso c'è uno scorrimento della lista dove ad ogni passo vengono sommati i soldi, una volta che viene vista tutta la lista viene inserita la somma totale e passata con una FullWrite. Nel secondo caso il BlockClient richiede una specifica transazione, dunque ancora una volta viene fatto lo scorrimento di tutta la lista se viene trovato il nodo viene inviato.

```

case 3:
    printf("E' stata selezionata la scelta 3\n");           /*scorre la lista dalla testa a partire da
    sleep(1);
    NodoN * ok2 = testa;
    FullRead(nuovo_socket, & numero_transazione, sizeof(int));

    while (numero_transazione > 0) {
        ok2 = ok2 -> next;
        FullWrite(nuovo_socket, ok2, sizeof(NodoN));
        numero_transazione--;
    }

    close(nuovo_socket);
    break;

case 4:
    printf("E' stata selezionata la scelta 4\n");           /*cerca l'indirizzo ip richiesto*/
    NodoN * ok3 = testa;
    FullRead(nuovo_socket, & ip, sizeof(ip));
    while (ok3 != NULL) {
        if ((strcmp(ok3 -> ip_mittente, ip) == 0) || (strcmp(ok3 -> ip_destinatario, ip) == 0)) {
            FullWrite(nuovo_socket, ok3, sizeof(NodoN));
            break;
        }
        ok3 = ok3 -> next;
    }
    close(nuovo_socket);
    break;
}

```

Nel terzo caso viene richiesto un numero dal quale visualizzare gli ultimi nodi.

Praticamente se ho 10 nodi nella lista e voglio vedere gli ultimi 6 nodi significa che la lista stamperà 10 9 8 7 6 5 4

Nel quarto caso viene letto l'ip che ha inviato il BlockClient e si richiede di ricerca una transazione partendo da questo ip con un strcmp ossia compara le string va a verificare se c'è una transazione che ha un ip mittente o destinatario a l'ip inviato dal BlockClient se è vero il BlockServer invierà questo nodo al BlockClient è lui lo stamperà.

BlockClient

```
Connect(listenfd, addr, sizeof(addr));
for (;;) {
    printf("Scegli un operazione\n");
    printf("1. Somma totale\n");
    printf("2. Transazione specifica da visualizzare\n");
    printf("3. Visualizzare gli ultimi N nodi\n");
    printf("4. Ip mittente o destinatario da trovare\n");

    scanf("%d", & scelta);
    if (scelta > 4) {
        printf("scelta sbagliata\n");
        exit(1);
    }
    FullWrite(listenfd, & scelta, sizeof(int));

    switch (scelta) {
        case 1:
            printf("\nQuesta è la somma totale \n");
            FullRead(listenfd, & totale, sizeof(int));
            printf("totale:%d\n", totale);

            break;

        case 2:
            printf("inserisci il numero della transazione che vuoi vedere\n");
            scanf("%d", & numero_transazione);
            FullWrite(listenfd, & numero_transazione, sizeof(int));
            FullRead(listenfd, nuovo_nodo, sizeof(NodoN));
            stampa(nuovo_nodo);

            break;

        case 3:
            printf("inserisci la transazione dalla quale vuoi vedere\n");
            scanf("%d", & numero_transazione);
            FullWrite(listenfd, & numero_transazione, sizeof(int));
            while (numero_transazione > 0) {
```

Dunque il blockClient semplicemente si connette al blockserver,entriamo in uno switch dove si sceglie quale opzione si vuole richiedere al blockserver viene eseguita la richiesta e chiuso il client.

Manuale Utente

Per compilare:

```
gcc NodoN.c -o nodo
./nodo
```

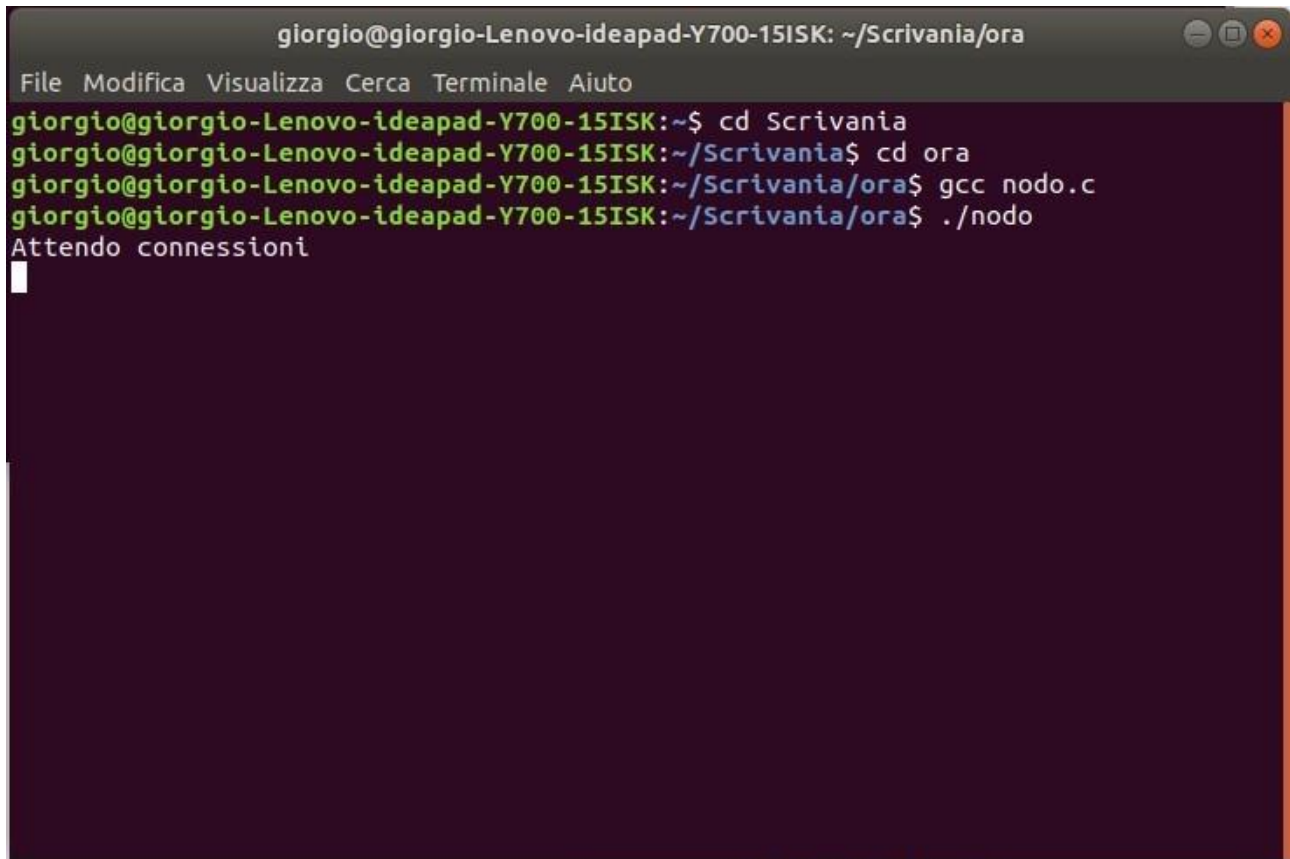
```
gcc BlockServer.c -o server
./server
```

```
gcc BlockClient.c -o client
```

./client

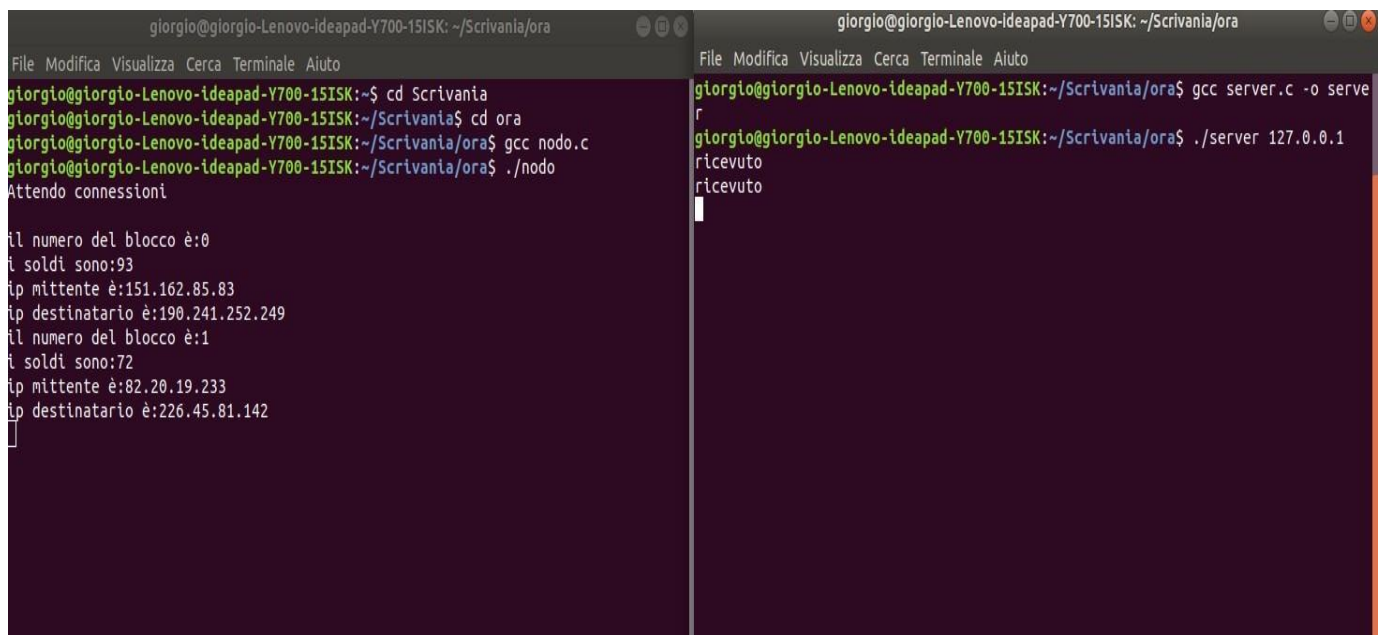
ESEMPIO DEL PROGRAMMA IN FUNZIONE

Qui è stato lanciato il **NodoN** che **attende una connessione**, quindi ./nodo



```
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK: ~/Scrivania/ora
File Modifica Visualizza Cerca Terminale Aiuto
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~$ cd Scrivania
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania$ cd ora
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania/ora$ gcc nodo.c
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania/ora$ ./nodo
Attendo connessioni
█
```

Qui è stato lanciato il **Blockserver** e inizia a ricevere nodi (./server)



```
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK: ~/Scrivania/ora
File Modifica Visualizza Cerca Terminale Aiuto
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~$ cd Scrivania
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania$ cd ora
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania/ora$ gcc nodo.c
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania/ora$ ./nodo
Attendo connessioni

il numero del blocco è:0
i soldi sono:93
ip mittente è:151.162.85.83
ip destinatario è:190.241.252.249
il numero del blocco è:1
i soldi sono:72
ip mittente è:82.20.19.233
ip destinatario è:226.45.81.142
█

giorgio@giorgio-Lenovo-ideapad-Y700-15ISK: ~/Scrivania/ora
File Modifica Visualizza Cerca Terminale Aiuto
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania/ora$ gcc server.c -o server
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania/ora$ ./server 127.0.0.1
ricevuto
ricevuto
█
```


Qui è stato lanciato il **BlockClient**, possiamo notare che nel momento della connessione il BlockServer stampa il messaggio “connesso”

```
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK: ~/Scrivanja/ora
File Modifica Visualizza Cerca Terminale Aiuto
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivanja/ora$ ./nodo
Attendo connessioni
il numero del blocco è:0
i soldi sono:93
ip mittente è:151.162.85.83
ip destinatario è:190.241.252.249
il numero del blocco è:1
i soldi sono:72
ip mittente è:82.20.19.233
ip destinatario è:226.45.81.142
il numero del blocco è:2
i soldi sono:21
ip mittente è:8.87.39.167
ip destinatario è:5.212.208.82
il numero del blocco è:3
i soldi sono:39
ip mittente è:117.27.153.74
ip destinatario è:237.88.61.106
il numero del blocco è:4
i soldi sono:39
ip mittente è:213.36.74.104
ip destinatario è:142.173.149.95

giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivanja/ora$ gcc server.c -o server
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivanja/ora$ ./server 127.0.0.1
ricevuto
ricevuto
ricevuto
ricevuto
ricevuto
connesso

giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivanja/ora
File Modifica Visualizza Cerca Terminale Aiuto
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivanja$ cd ora
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivanja/ora$ gcc client.c -o client
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivanja/ora$ ./client 127.0.0.1
Scegli un operazione
1. Somma totale
2. Transazione specifica da visualizzare
3. Visualizzare gli ultimi N nodi
4. Ip mittente o destinatario da trovare
```

In questo esempio il BlockClient richiede la somma totale della transazioni e il BlockServer fa la somma e invia il risultato al BlockClient.

Quando il blocco Client semplice non è sufficiente, si sceglie quale opzione si vuole e si chiama il blocco Client

Per compilare:

```
File Modifica Visualizza Cerca Terminale Aiuto
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania$ gcc
r
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania$ ./se
ricevuto
ricevuto
ricevuto
ricevuto
ricevuto
ricevuto
connesso
ricevuto
E' stata selezionata la scelta 1
[ ]

giorgio@giorgio-Lenovo-ideapad-Y700-15ISK: ~/Scrivania/o

File Modifica Visualizza Cerca Terminale Aiuto
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~$ cd Scrivania
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania$ cd ora
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania/ora$ gcc
t
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania/ora$ ./cl
Scegli un operazione
1. Somma totale
2. Transazione specifica da visualizzare
3. Visualizzare gli ultimi N nodi
4. Ip mittente o destinatario da trovare
1

Questa è la somma totale
totale:264
giorgio@giorgio-Lenovo-ideapad-Y700-15ISK:~/Scrivania/ora$ [ ]
```