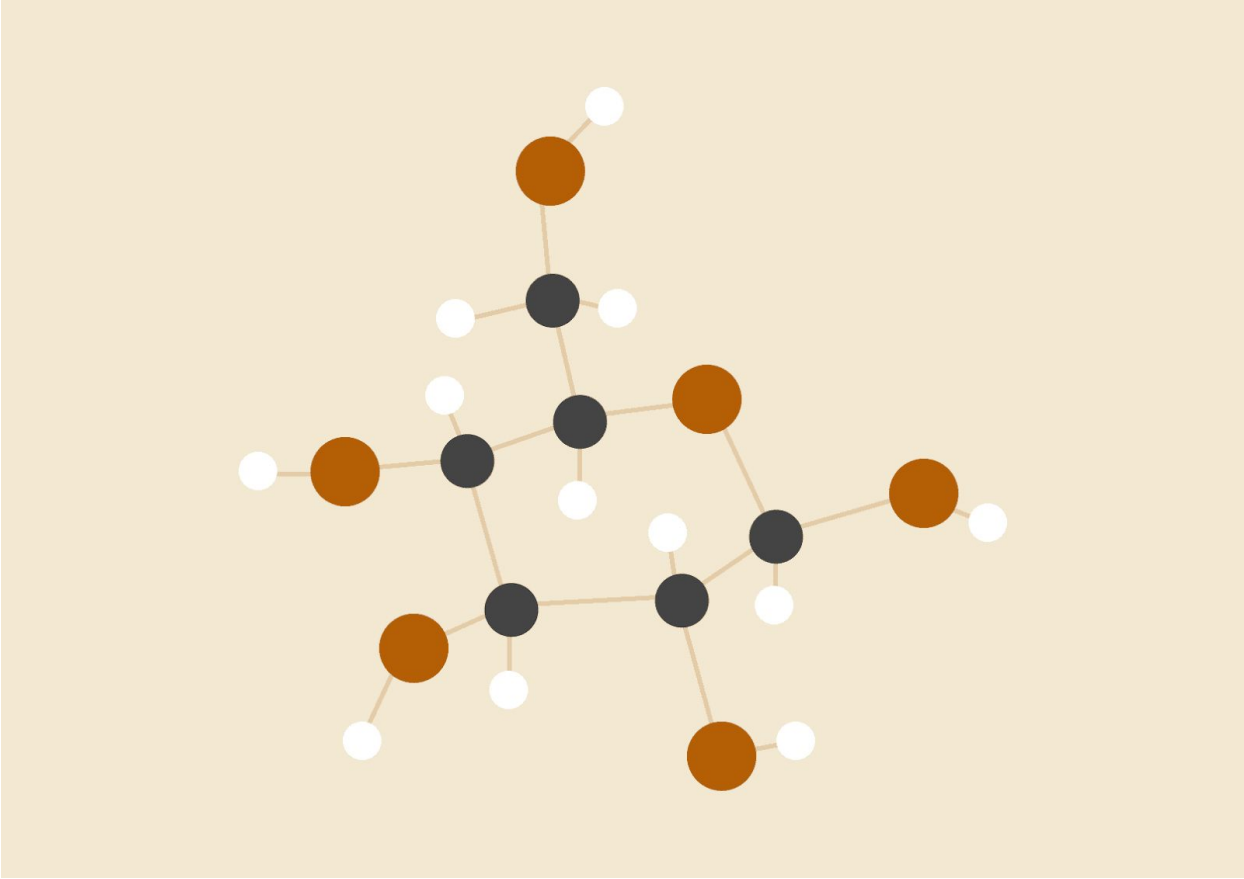
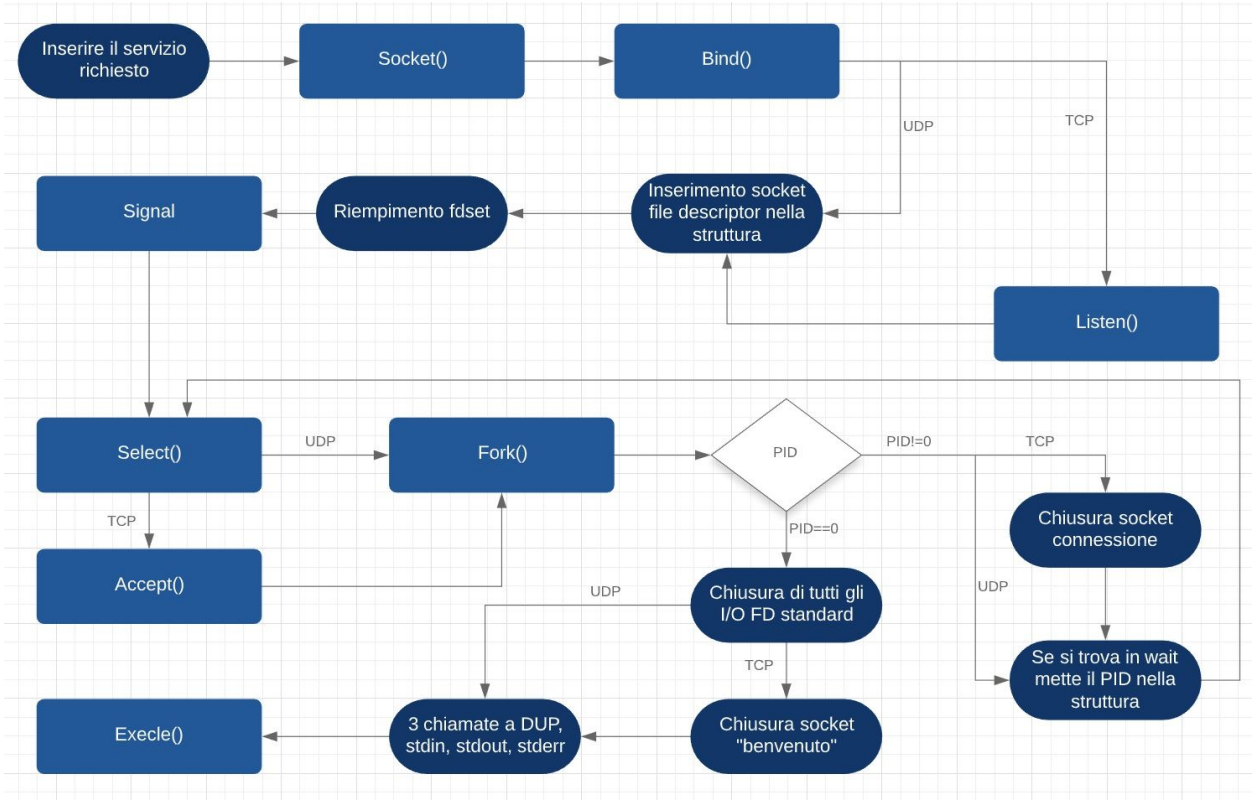
RELAZIONE ASSIGNMENT 2 DI RETI DI CALCOLATORI

Ismam Abu, Marco Hu, Zheliang Zhu



TASK 1

In seguito il diagramma che riassume il funzionamento:



TASK 2

PROGETTAZIONE DEL SUPERSERVER:

Per prima cosa vengono salvati i parametri del servizio nella struttura dati indicata in figura.

Vengono fatte chiamate le funzioni di socket() e bind(), nel caso la connessione sia TCP c'è anche listen()

Il socket file descriptor viene inserito nella struttura dati, la select viene riempita con i socket file descriptor.

Il file superserver.txt contiene al suo interno il nome di ogni servizio, viene letto e vengono eseguite le operazioni.

Viene invocata la signal e poi la select() per la lettura, nel caso il socket sia TCP viene eseguito anche accept(), successivamente viene eseguita una fork(), questa funzione genera un processo figlio che mette in esecuzione il servizio e un processo padre che svolge la funzione di superserver, il padre quindi è in un ciclo infinito per vedere se si sta attivando una connessione.

Il figlio, in caso di connessione TCP chiude la socket di benvenuto e chiama execle() per eseguire il servizio.

In generale sono state seguite le linee guida fornite dalla consegna, si è utilizzato lo scheletro aggiungendo le parti mancanti.

COME COMPILARE IL SUPERSERVER:

Per semplicità abbiamo scritto un file Makefile con al suo interno le regole di compilazione che comprendono sia il superserver che i servizi.

(se si scrive “make makec” si esegue la compilazione dei 4 file forniti, per toglierle “Make maked”).

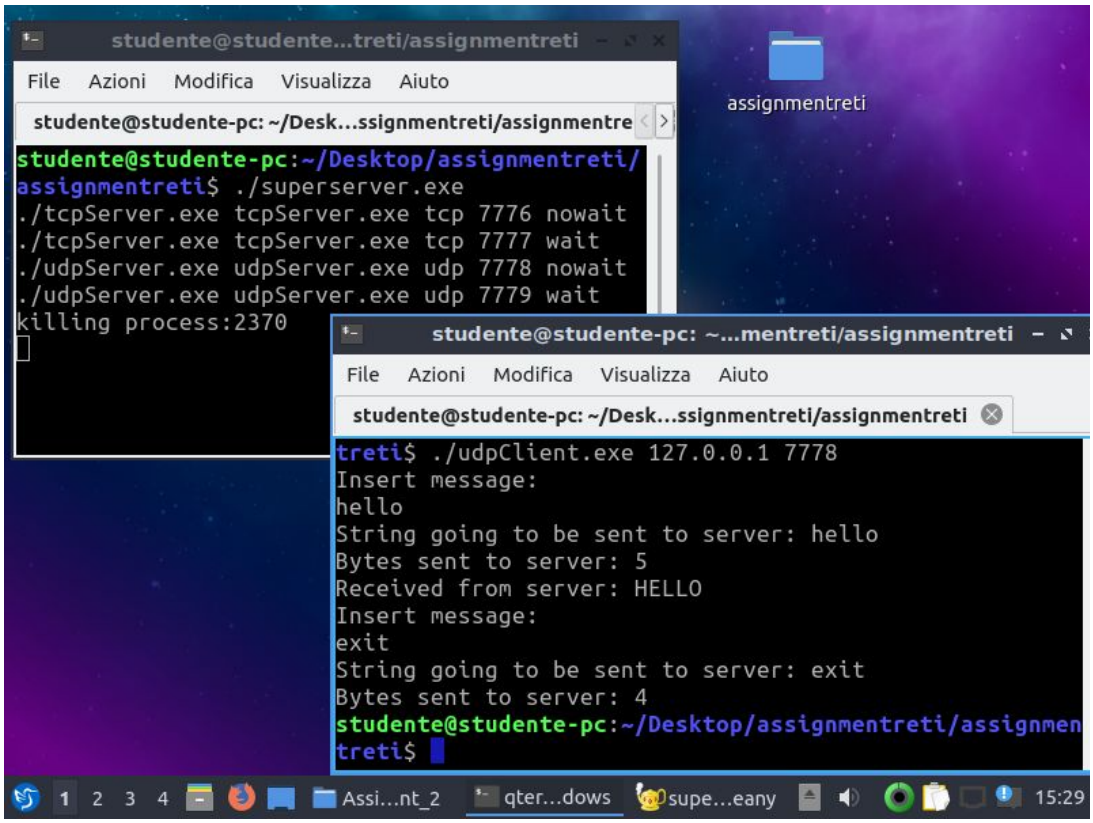
La struttura dati concepita per salvare i parametri è la seguente

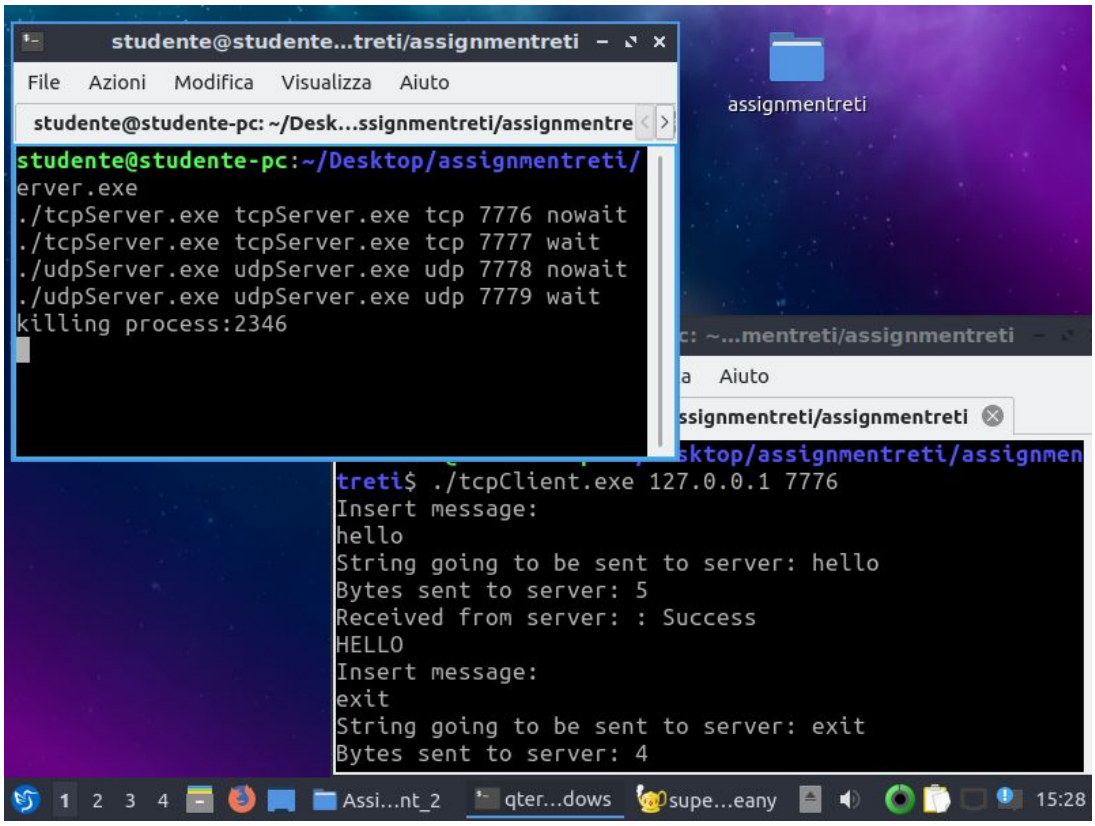
```
//Service structure definition goes here
typedef struct
{
    char TransportProtocol[4];  //'tcp' if protocol is TCP, 'udp' if protocol is UDP
    char serviceMode[7];        //'wait' if service is concurrent, 'nowait' otherwise
    char port[6];           //port on which service is available
    char CompleteName[50];      //complete name of the service, including full path
    char Name[20];          //name of the service
    int SocketDescriptor;       //socket descriptor associated to the port
    int PID;            //process ID
} serviceInfo;

serviceInfo si[FDSIZE];
```

TEST REPORT:

Con il client in wait il superserver gestisce i client uno alla volta, per poter gestire un nuovo client bisogna che quello precedente sia stato gestito.

Quando il client è in non wait il superserver riesce a gestire più client alla volta in quanto non deve aspettare che un client finisca per poter gestire il successivo.


studente@studente...treti/assignmentreti
File Azioni Modifica Visualizza Aiuto
studente@studente-pc: ~/Desk...ssignmentreti/assignmentre
studente@studente-pc:~/Desktop/assignmentreti/
assignmentreti$ ./superserver.exe
./tcpServer.exe tcpServer.exe tcp 7776 nowait
./tcpServer.exe tcpServer.exe tcp 7777 wait
./udpServer.exe udpServer.exe udp 7778 nowait
./udpServer.exe udpServer.exe udp 7779 wait
killing process:2370
assignmentreti
studente@studente-pc: ~...mentreti/assignmentreti
File Azioni Modifica Visualizza Aiuto
studente@studente-pc: ~/Desk...ssignmentreti/assignmentreti
treti$ ./udpClient.exe 127.0.0.1 7778
Insert message:
hello
String going to be sent to server: hello
Bytes sent to server: 5
Received from server: HELLO
Insert message:
exit
String going to be sent to server: exit
Bytes sent to server: 4
studente@studente-pc:~/Desktop/assignmentreti/assignmen
treti$
1 2 3 4 Assi...nt_2 qter...dows supe...eany 15:29


studente@studente...treti/assignmentreti
File Azioni Modifica Visualizza Aiuto
studente@studente-pc: ~/Desk...ssignmentreti/assignmentre
studente@studente-pc:~/Desktop/assignmentreti/
erver.exe
./tcpServer.exe tcpServer.exe tcp 7776 nowait
./tcpServer.exe tcpServer.exe tcp 7777 wait
./udpServer.exe udpServer.exe udp 7778 nowait
./udpServer.exe udpServer.exe udp 7779 wait
killing process:2346
assignmentreti
Aiuto
treti$ ./tcpClient.exe 127.0.0.1 7776
Insert message:
hello
String going to be sent to server: hello
Bytes sent to server: 5
Received from server: : Success
HELLO
Insert message:
exit
String going to be sent to server: exit
Bytes sent to server: 4
1 2 3 4 Assi...nt_2 qter...dows supe...eany 15:28