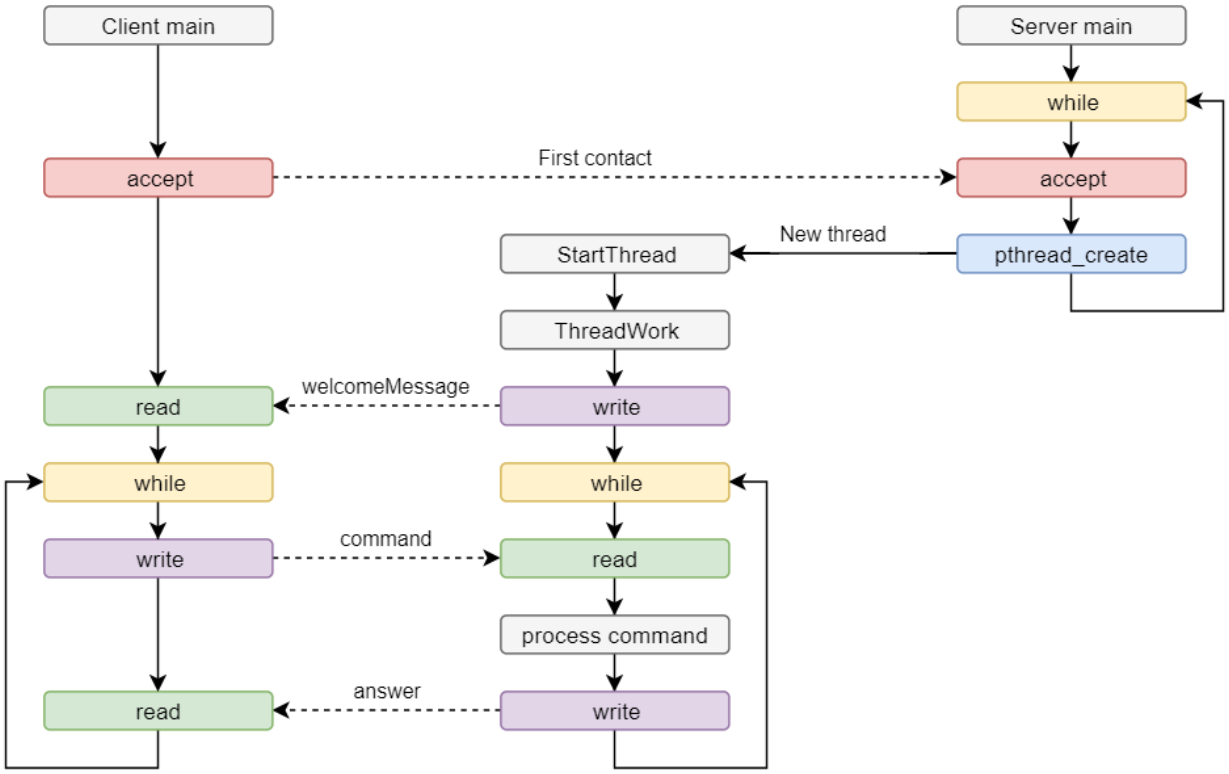
1. Introducere

Aplicatia myRPC este un sistem creat format dintr-o componenta client si o componenta server, creat dupa modelul *Remote procedure call*, care are la baza protocolul TCP. Aceasta permite clientului sa se conecteze la un server si sa apleleze un numar de comenzi suportate de server-ul respectiv, urmand ca serverul avand sa transmita inapoi catre client, un raspuns in format text.

2. Tehnologii utilizate

Cele doua componente, client si server, realizeaza comunicarea prin intermediul protocolului *Transmission Control Protocol* sau TCP. Am ales acest protocol deoarece protocolul TCP este orientat pe conexiune, asigurand astfel integritatea mesajului transmis si certitudinea ca el ajunge la destinatar, lucruri esentiale in comunicarea dintre cele doua componente. Desi TCP are o viteza de transmisie mai mica fata de protocolul *User Datagram Protocol* sau UDP, date fiind serviciile oferite de aplicatie, aceasta nu are nevoie de o viteza de transmisie mai mare. Un alt avantaj important al protocolului TCP este capabilitatea de a detecta si eventual corecta erorile aparute in mesaj in urma transmisiei.

3. Arhitectura aplicatiei



Componenta client lucreaza pe un singur fir. Aceasta face contactul cu server-ul prin functia accept, iar apoi asteapta de la server mesajul de primire. Acest pas este obligatoriu iar in cazul in care se doreste ca serverul sa nu trimita un mesaj de primire, trebuie ca acesta sa trimita un intreg cu valoarea 0, mai precis 4 octeti cu valoarea 0. Dupa receptionarea mesajului de primire clientul incepe bucla principala in care acesta asteapta introducerea unei comenzi de la tastatura (*stdin*), dupa care trimite comanda si asteapta raspunsul din partea server-ului. Singura comanda care este analizata local este cea pentru inchiderea aplicatiei. In urma acestei comenzi, clientul trimite un mesaj care notifica server-ul asupra deconectarii sale.

Serverul permite conectarea si servirea mai multor clienti simultan, intrucat acesta utilizeaza mai multe fire de executie, lucru facilitat de libraria *pthreads*. Dupa ce se efectueaza primul contact prin functia *accept*, server-ul creaza un nou fir de executie care porneste din functia *StartThread* si primeste ca argument datele necesare pentru a comunica cu clientul, mai precis descriptorul asociat. Activitatea principala se executa in functia *ThreadWork*. Server-ul trimite mai intai un mesaj de primire catre client, dupa care incepe bulca principala in care asteapta ca clientul sa trimita comenzi pe care acesta le parseaza, le executa, trimite raspunsul inapoi catre client si in final revine la asteptarea de noi comenzi. La deconectarea clientului, fie ea intentionata sau nu, firul de executia dedicat clientului respectiv se termina.

Trimiterea mesajelor se realizeaza dupa protocolul urmator: se trimite un intreg pe 4 octeti care semnifica lungimea mesajului in octeti care urmeaza sa fie transmis; apoi se transmite mesajul. La receptionarea mesajului, se citesc cei 4 octeti pentru aflarea lungimii mesajului si apoi se citeste numarul corespunzator de octeti.

4. Detalii de implementare

Din punctul de vedere al networking-ului, aplicatia respecta o formula standard, de baza. Particularitatile se regasesc in modul de indexare al comenzilor, procesarea unelor dintre acestea si tratarea deconectarii premature a clientilor.

- Indexarea comenzilor

Pentru indexarea comenzilor am folosit o enumerare in care am atribuit fiecarei comenzi o valoare intreaga. Pe langa aceasta, exista un vector care retine literalii corespunzatori fiecarei comenzi. Acest mod de a reprezenta fiecare comanda faciliteaza identificarea acestora dar si adaugarea ulterioara de noi comenzi.

```
// Enumerare comenzi
enum Command{
    CMD_UNDEFINED = 0,
    CMD_HELP = 1,
    CMD_QUIT = 2,
    CMD_TIME = 3,
    CMD_MYIP = 4,
    CMD_PING = 5,
    CMD_FACT = 6
};
```

```
// Vector literali comenzi
char * commandStr[]{
    "undefined",
    "help",
    "quit",
    "time",
    "myip",
    "ping",
    "fact"
};
```

```
// Identificarea comenzii
Command GetCommand(char * buffer)
{
    Command command = CMD_UNDEFINED;                //  se asuma comanda
                                                    // nedefinita
    for(int i = 1; i < commandCount; i++)
        if(StartsWith(buffer, commandStr[i]) &&     //  se verifica daca
           iswspace(buffer[strlen(commandStr[i])])) // comanda incepe cu
        {                                           // literalul sau
            return (Command) i;
        }

    return command;
}
```

Comanda aleasa se proceseaza in functia *ThreadWork* printr-o bucla de tip *switch* dupa functia aleasa.

- Procesarea comenzii *ping*

Comanda *ping* se executa prin intermediul functiei *popen*. Aceasta permite testarea conexiunii catre o adresa aleasa de client dar si intre server si client daca acesta introduce ‘myself’ ca argument in locul adresei.

```
// Implementare comanda ping
int PingCommand(char * arg, char * answer, char * clAddr)
{
    char * cmd = new char[128];

    if (arg == NULL)                                        //  verificarea
    {                                                       // argumentului
        sprintf(answer, "You need to specify an address\n");
        return 0;
    }

    if(StartsWith(arg, "myself") && iswspace(arg[6]))
    {
        sprintf(cmd, "ping -c 1 %s", clAddr);
    }                                                   //  formarea comenzii
    else                                                // pentru popen
    {
        sprintf(cmd, "ping -c 1 %s", arg);
    }
    FILE * p = popen(cmd, "r");    // apelare popen

    char * line = new char[128];

    while(fgets(line, 512, p))
    {                                                       //  citirea
        strcat(answer, line);                               // rezultatelor
    }
    delete [] cmd;
    delete [] line;
    cmd = NULL;
    line = NULL;
    return 0;
}
```

- Tratarea deconectarii premature a clientului

Server-ul verifica in permanenta integritatea raspunsului primit de la client. Acesta face verificari asupra numarului de octeti primit dar si asupra mesajului in sine. Daca se constata o problema acesta va raporta in acest lucru si va inchide firul de executie si implicit conexiunea.

```
// Verificare conexiune
while(!quit)
{
    printf("[Thread #%d][%s] Waiting for user input\n", tdL.idThread, clAddr);
    read(tdL.cl, &commandLength, sizeof(int));
    if(commandLength == 0)
    {
        printf("[Thread #%d][%s] Received null message from client. Closing
                connection.\n", tdL.idThread, clAddr);
        quit = 1;
    }
    else
    {
        . . . // citirea comenzii

        switch (command) {

            . . . // executare comenzi

            default: {
                    char *c = strtok(commandBuffer, " \t\n\v\f\r");
                    if (c == NULL)    // verificarea mesajului nul
                    {
                        printf("[Thread #%d][%s] Lost connection with user.\n",
                                tdL.idThread, clAddr);
                        quit = 1;
                    }

                    . . . // tratarea comenzii nerecunoscute

                    break;
            }
        }
    }
    . . .
}
```

5. Concluzii

Aplicatia ofera clientilor un mod simplu de a comunica cu o entitate ce furnizeaza diferite servicii. Adaugarea de noi comenzi consta exclusiv in modificarea componentei server, astfel nu este nevoie de nici un efort din partea clientului in scopul utilizarii de noi comenzi.

Pe langa dezvoltarea functionalitatii server-ului prin adaugarea de noi comenzi, o alta imbunatatire care poate fi adusa aplicatiei ar fi utilizarea unui fisier de configurare sau eventual mai multe comenzi cu caracter local care sa permita modificarea a diferiti parametri ai programului.

Posibilitatea de a transfera fisiere intre client si server este si ea o imbunatatire care ar creste drastic domeniul de servicii pe care le poate oferi un server.

6. Bibliografie

- https://profs.info.uaic.ro/~eonica/rc/index.html
- https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php
- https://linux.die.net/man
- http://www.cplusplus.com