ТЕХНИЧЕСКИ
УНИВЕРСИТЕТ-СОФИЯ
ФКСТ/КСИ

КУРСОВА РАБОТА
ПО
СИСТЕМНО ПРОГРАМИРАНЕ

на тема:

Създаване и моделиране на ,,К списък”

Изготвил:
Валентин Георгиев Александров,
Фак.No 121216074, гр.46
Дата: 19.05.2018

Одобрил:.....
/ас. Д . Андр

СЪДЪРЖАНИЕ:

1. Анализ на изготвеното приложение

- Задание

Да се моделира ,,K списък” със следните качества – той притежава същите качества както стандартния едносвързан списък, но за всеки елемент освен връзка към следващ елемент се пази и връзка към K-ти елемент напред. По този начин може да се осъществи евентуално по-бързо търсене. Например при к=10, ако търсите 62-ри елемент стандартно ще направи 62 операции:

1 – 2 – 3 – … - 61 – 62

Докато използвайки модифицираната функция get(head, 62) търсенео ще бъде с 9 операции:

1 – 10 – 20 – 30 – 40 – 50 – 60 – 61 – 62

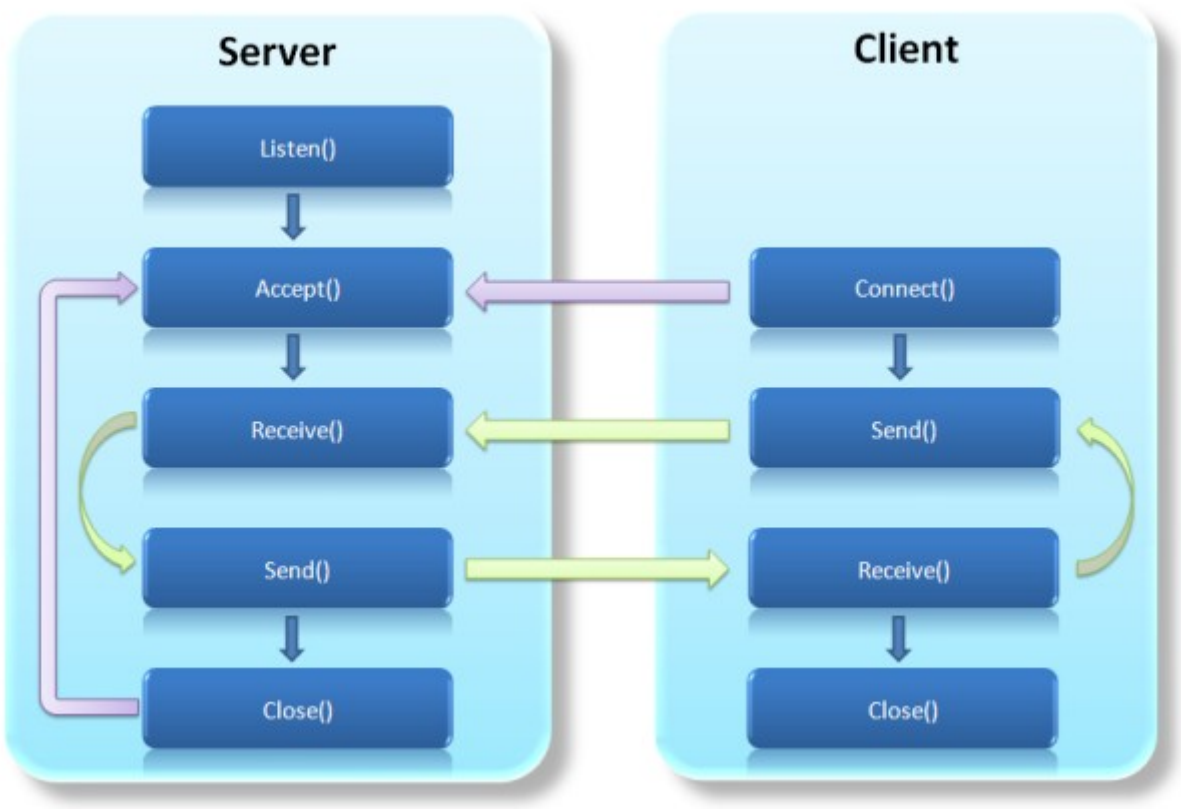
Направете main метод, в който зареждате списък от бинарен файл с няколко хиляди елемента – може да са приозволни числа. Проверете с колко по-бърза ще бъде новата функция.

- Интерпретация

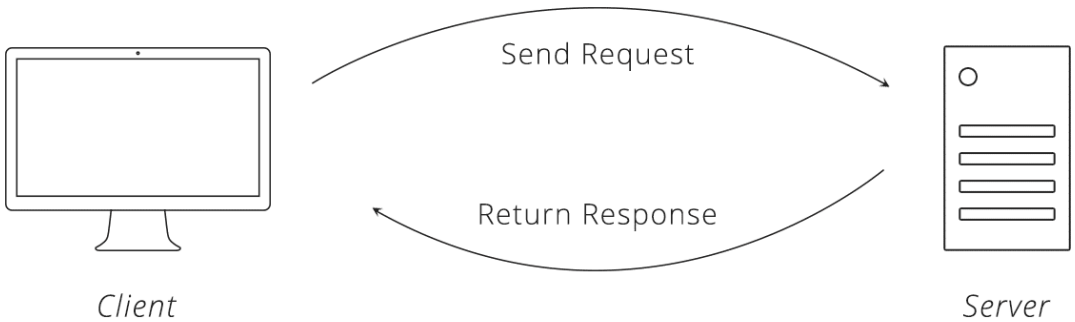
Трябва да се създаде приложение, което да чете числа от бинарен файл и да ги записва в едносвързан списък. За по-бързо четене този списък е модифициран и има указател към K-тия елемент. Трябва да се направи клиентска страна, която да праща заявка, която зазарежда данните от файл на сървъра с структура от данни – едносвързан списък. Освен това са неоходими още две заявки, които се изразяват в търсенето на даден елемент от списъка по два начина – елемент по елемент или през K – елемента. Комуникацията между сървъра и клиента се осъществява чрез сокети. Сървъра ще се занимава с извличането на данни и търсенето из тях, а клиента просто ще визуализира отговора на своите заявки към сървъра. На клиента ще се пратят всички елементи през които е било необходимо да се премине за да се намери търсения елемент и по това ще си проличи колко ефикасен модифицирания метод за търсене през K-тия елемент.

2. Функционално описание на приложението

Диаграма, показваща архитектурата на комуникиращите звена на приложението (клиет и сървър)

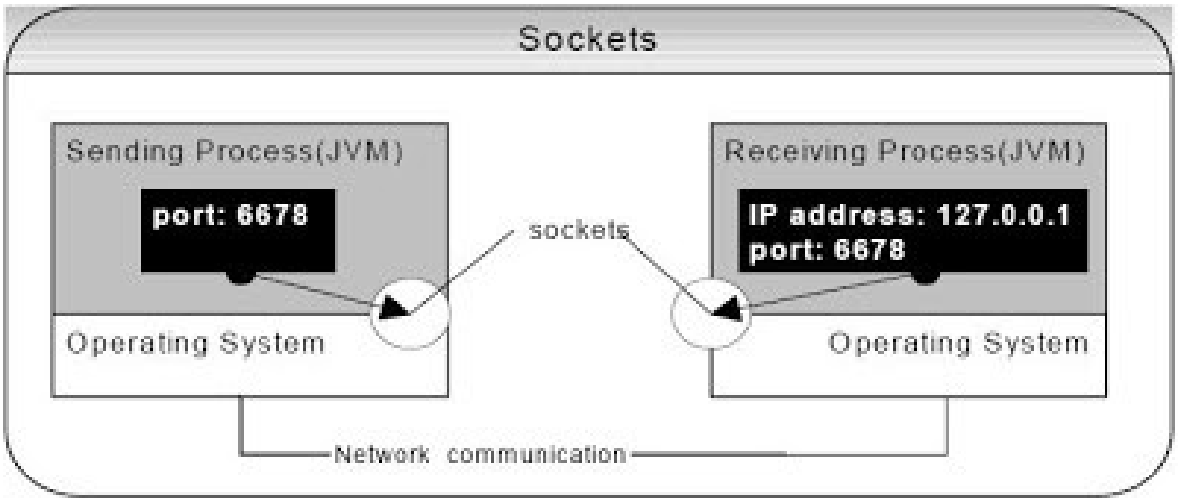


Сървърът слуша за на своя порт за клиенти, които може да се свържат към него. След успешно свързан клиент, се пуска нова нишка, която ще работи само с единия клиент. Тази тактука позволява, в този момент, сървъра да почне да се свързва с още един клиент, който ще бъде обслужен на отделна нишка. В приложението клиента и сървъра са два процеса, които си комуникират чрез сокети. Процесът може да бъде описан по следния начин:



И това се повтаря до условие за прекъсване на комуникацията.

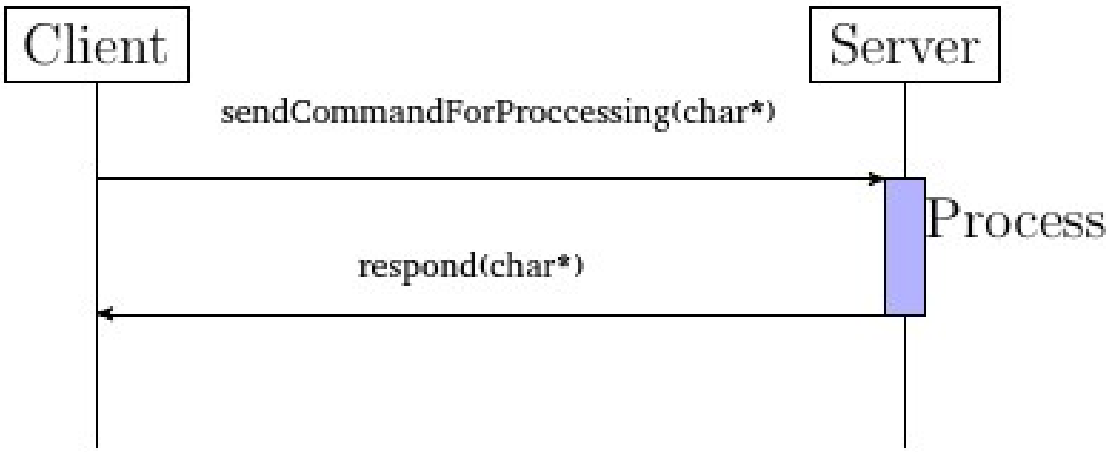
В контекста на това приложение тази комуникация е реализира на по следния начин:



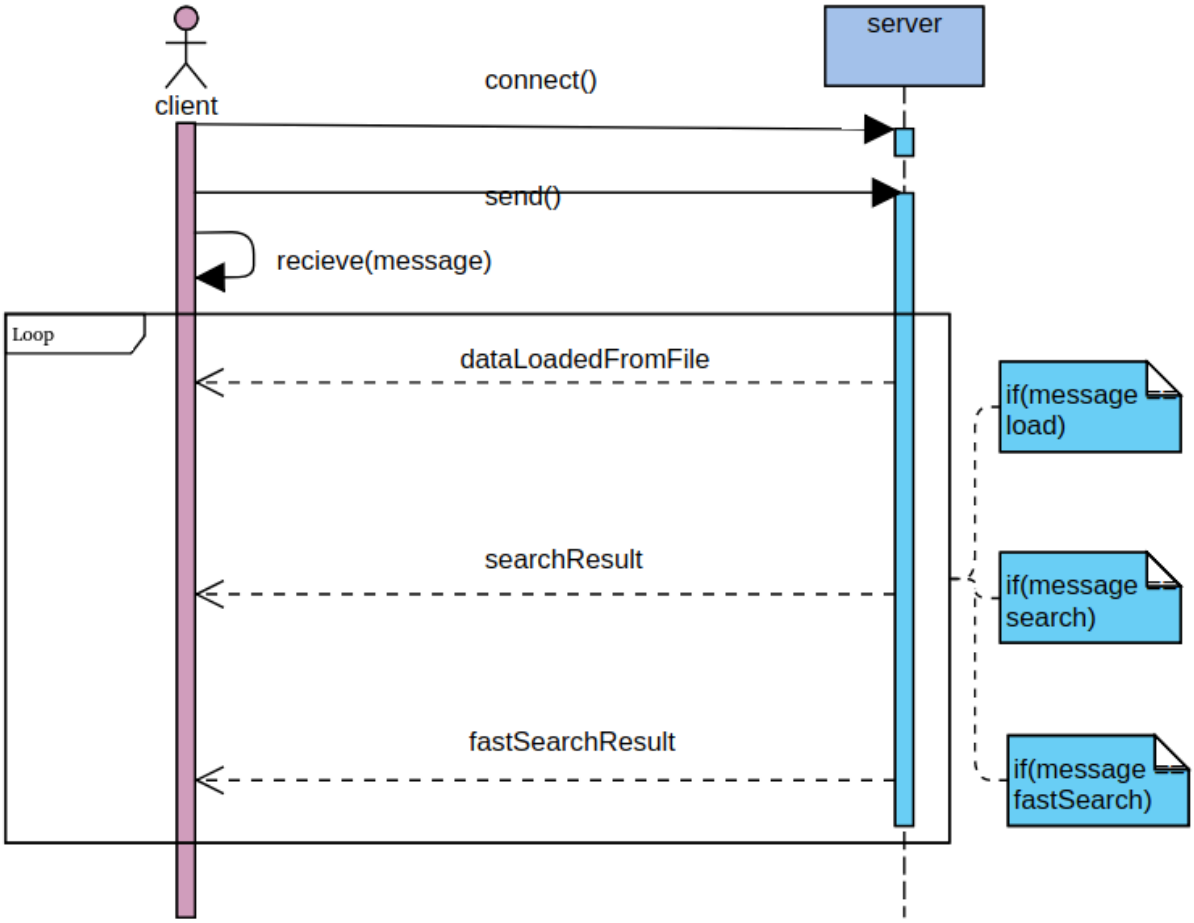
Клиентския процес се свързва към сървъра на даден IP адрес със съответен порт.
ВАЖНО: Портът трябва да е свободен!

Диаграми на последователностите:

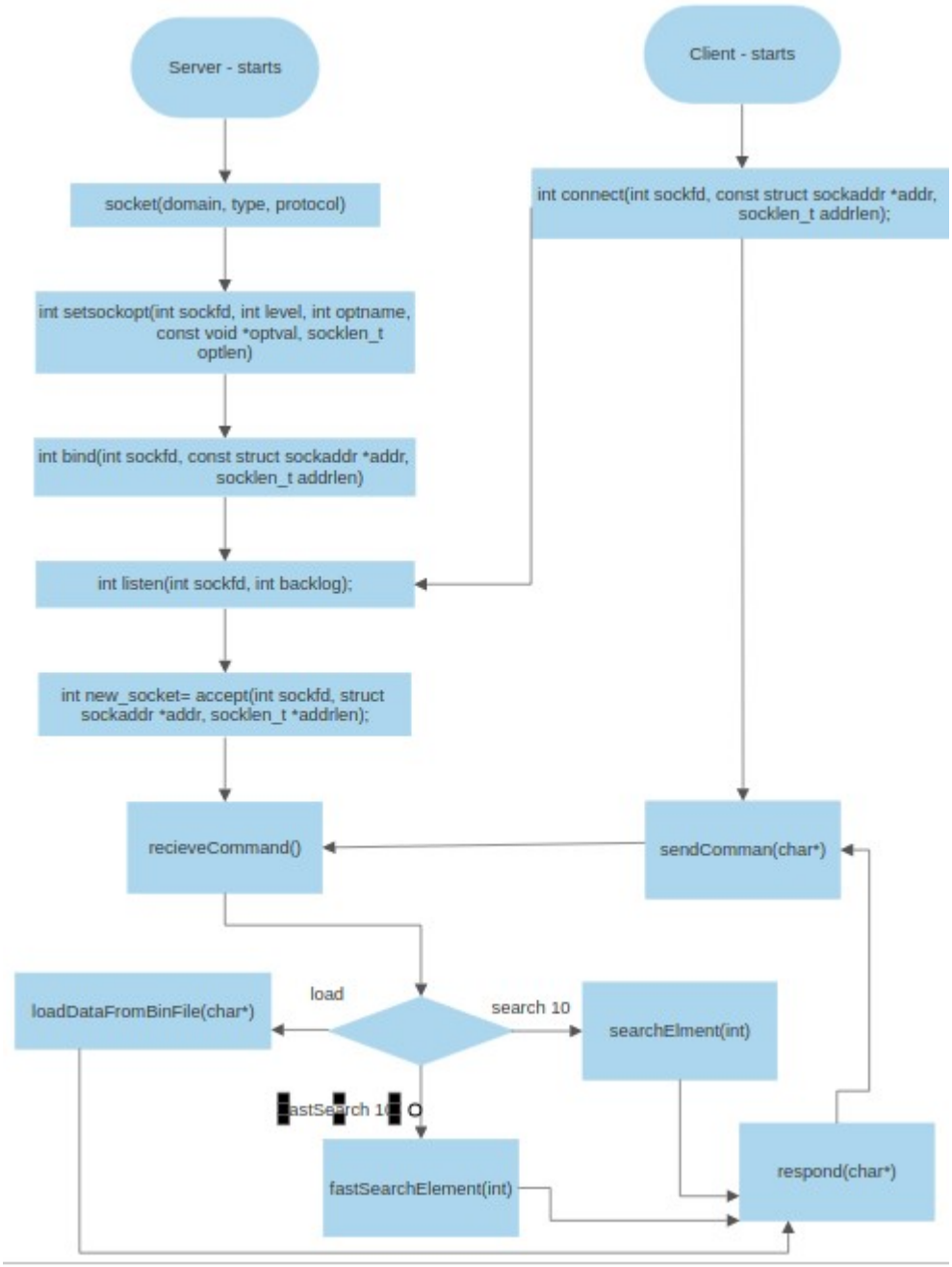
-Образна комуникация между клиента и сървъра



-Комуникация в скелета на програмата



UML workflow diagram:



Приложение работи спрямо изпратените от клиента команди. Те са:
- Извлечи данните от файл с дадено име.
- Потърси число в структурата от данни.
- Потърси по-бързата процедура число от списъка.

Клиента получава съобщение от сървъра, в което се съдържда търсения ресурс.

3.Изпълнение на функционалностите

loadDataFromBinFile(char*) - функция, която прочита числата от бинарен файл с дадено име и ги пълни в едносвързан списък със структура описана по условието.

searchElement(int) – функция, която минава през всичките елементи последователно и записва всяко число, през което е минала вече докато не намери търсеното число.

fastSearchElement(int) – функция, която работи подобно на горната, обаче обхожда списъка през K-тия елемент.

sruct LinkedList* genareteNewLinkedList(struct LinkedList *head) – функция, която създава нов едносвързан списък, за да може в него да се напълнят данните от съответния избран бинарен файл.

4. Експериментални данни.

-Стартиране на работата

Compiling:

```
gcc client.c -o client
gcc server.c -o server
```

Run:

```
./client
./server
```