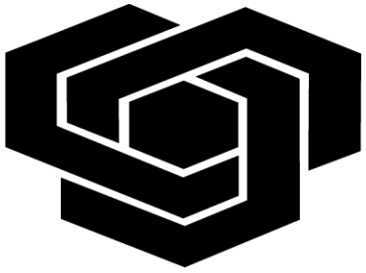


ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Факултет по комптютърни системи и технологии

Проект за оценка по системно програмиране

Информация за студента:

Име: Ясен Ивов Стоилов

Курс: III

Група.: 46

Ф.н.: 121216081

Дата: 22.05.2019

Проверил:

Д. Андреев

1. Анализ на изготвеното приложение

Задание:

Дискотеките на Студентски град се пръскат по шевовете от светло до светло. Разбира се, всяко себеуважаващо се заведение предлага меню от поне десет артикула, имащи име, цена, количество. Ако количеството на даден продукт се изчерпи, то той бива изтриван. Да предположим, че всяка вечер се пълнят по двадесет маси, които правят различни поръчки. Сервитьорите могат постоянно да гледат коя маса какви поръчки е направила. Накрая на вечерта, преди спирането на соофтуера, се записва оборотът затози ден. Понякога идват данъчни и им се дава възможност да поглеждат датата и дневния оборот.

Целта на приложението е да се симулира работата на нощно заведение. Ако приемем, че от една страна е персоналът на заведението, от друга клиентите, настанени по маси(или под тях), а от трета е някакъв данъчен инспектор, който периодично минава да отчете оборота, то тези ‘участници’ могат да се моделират като процеси, които взаимодействат помежду си. Един начин за реализация е под формата на връзка между клиент и сървър посредством TCP. В случая ролята на сървъра би могло да играе заведението, а клиентите на заведението да са... клиенти и да изпращат заявки към сървъра (поръчки). Данъчните също могат да се причислят към клиентите, защото и те в крайна сметка искат достъп до ресурс на сървъра (оборота).

2. Функционално описание и изпълнение на функционалностите

Програмата се състои от два отделни процеса, които стартират в отделни терминали. Началото на изпълнението и на сървъра, и на клиента е съсредоточено в това да се направи връзка с другия чрез създаване на socket и свързване към избраните IP и порт.

В кода на сървъра се инициализира и една статична структура от данни, представляваща наличните продукти. Всеки продукт се характеризира с име, цена и количество.

Когато връзката между клиента и сървъра е осъществена, клиентът започва да прави заявки към сървъра. На случаен принцип се избира типа на заявката – поръчка на артикул за маса, данъчна проверка, или поръчка на песен.

При направа на поръчка за даден артикул, на случаен принцип се избира количество и се формира заявка, която се подава към сървъра. При обработката на заявката, сървъра трябва да провери дали артикулът е наличен, дали се разполага с поисканото количество (ако е поръчано повече от наличното, наличното се предоставя на клиента и сървърът уведомява, че съответният продукт вече е изчерпан). Задача на сървъра също така е да преизчисли оборота.

В случай на данъчна проверка клиентът изпраща малко по-различна заявка, а сървърът връща информация за оборота, както и точни дата и час на посещението.

Масите имат и възможност да поръчат песен, като за това бива изпращана друга заявка, която сървърът обработва и връща отговор дали молчата е изпълнима. Ако да, то в отговора се съдържа и сумата бакшиш, която клиентът е оставил, което се отразява на оборота.

Работата на заведението спира, когато са изчерпани всички продукти за вечерта.

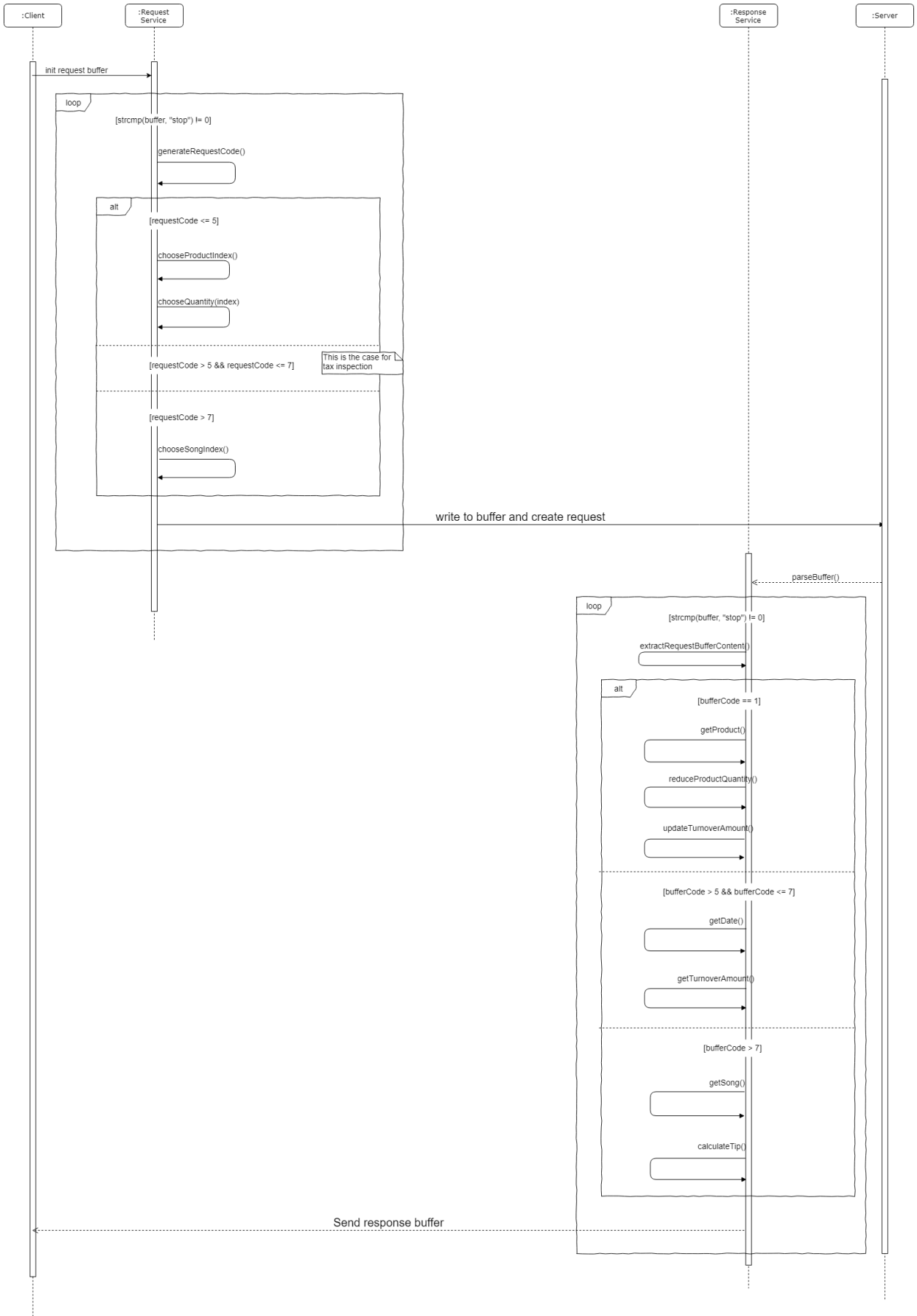
Клиентът непрекъснато изпълнява функцията void func(int sockfd); Функцията използва локален char буфер, който служи за комуникация със сървъра. Посредством буфера, клиентът черпи информация относно случващото се с поръчките му. Функцията също така използва случаен принцип за избиране на следващата заявка към сървъра, като конкретна песен се избира по индекс. Конструирането на заявката се извършва чрез библиотечни функции, а комуникацията със сървъра – със системните call-ове read, write, а и с още една функция – bzero за изчистване на буфера.

Сървърът също използва функция void func(int sockfd); Той още така пази инициализиран в началото масив от структури, съдържащи информация за продукти, както и един масив с имена на песни. В сървъра са декларирани и глобални променливи за броя на артикулите (за да се знае кога сървърът да спре) и оборота. Сървърът отново обработва заявките с помощта на char buffer[] като записването и извличането на информацията се улеснява с функциите strtok() и sprint(). Комуникацията с другата страна също е чрез системни call-ове read и write. По-добра представа може да се добие от прикачената по-долу sequence диаграма

3. Експериментални данни

Данните за артикулите в сървъра са подходящо инициализирани, за да могат да се използват и при пускане на двата процеса (клиента и сървъра) и да се постигне симулация.

4. Изходен код на приложението - https://github.com/yasenst/SPR


:Client
:Request Service
:Response Service
:Server
init request buffer
loop
[strcmp(buffer, "stop") != 0]
generateRequestCode()
alt
[requestCode <= 5]
chooseProductIndex()
chooseQuantity(index)
[requestCode > 5 && requestCode <= 7]
This is the case for tax inspection
[requestCode > 7]
chooseSongIndex()
write to buffer and create request
parseBuffer()
loop
[strcmp(buffer, "stop") != 0]
extractRequestBufferContent()
alt
[bufferCode == 1]
getProduct()
reduceProductQuantity()
updateTurnoverAmount()
[bufferCode > 5 && bufferCode <= 7]
getDate()
getTurnoverAmount()
[bufferCode > 7]
getSong()
calculateTip()
Send response buffer