

Документация на курсов проект по „Мрежово програмиране“

Тема: Редактор на изображения

Изработил: Цветомир Стайков - 1MI0800469

Специалност: Компютърни науки 3 курс

14 Декември 2025

Съдържание

Въведение и цели.....	2
Технологии.....	2
Разработка.....	2
Компилиация.....	2
Архитектура на проекта.....	2
Свързване.....	2
Клиент.....	2
Сървър.....	3
Алгоритми.....	4
Мрежова комуникация.....	4
Обработка на грешки и сигурност.....	5
Скорост на работа.....	6
Пример.....	6
Ограничения и бъдещо развитие.....	6
Краен резултат.....	7

Въведение и цели

Проектът представлява програма от две части – клиент и сървър. Клиентът има интерфейс с команди (CLI). Предоставени са няколко основни команди. Процесът на работа е линеен – зареждане на снимка, избиране на вид обработка, изпращане на снимка. Сървърът показва какво се случва в момента – няма възможност за вход. Той приема снимката и обработва по даден алгоритъм, след което връща резултата.

Технологии

Разработка

Проектът е реализиран изцяло на C++ на операционна система Linux Fedora 43. Използват се стандартните библиотеки.

Тестван е само на Linux Fedora 43.

Програмата е разделена на няколко отделни файла, като всеки отговаря за дадена цел. Някои от тях се използват както от сървъра, така и от клиента.

Компиляция

За компилиране може да се използва g++ компилаторът, но е предоставен CMakeLists.txt файл за компилация с CMake. Примерни команди в bash:

```
mkdir build  
cd build  
cmake ..  
make
```

Архитектура на проекта

Проектът работи специфично с файлове BMP, като поддържа версии от 2 до 5.

Свързване

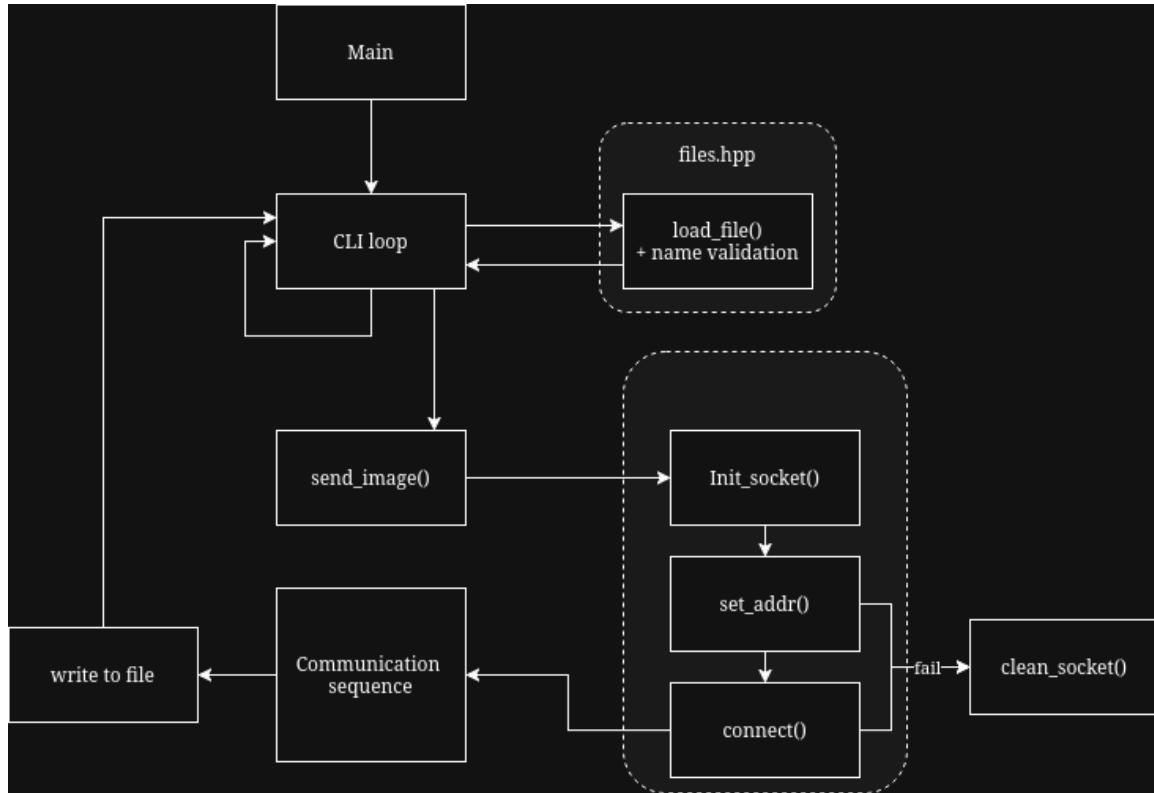
За комуникация между сървъра и клиента е избран TCP протоколът, поради нуждата от надежна връзка без загуби на информация. Комуникацията се осъществява по модел много клиенти - един сървър. Сървърът отделя нишка за всеки клиент.

Клиент

Клиентът има два основни момента – зареждане на изображение и изпращането му.

При зареждането се прави кратка проверка за името на файла и след това за съдържанието на header областта. Ако форматът съвпада, продължаваме.

При изпращането на данни свързването към сървъра става единствено, когато сме готови да изпратим снимката. След като получим данните за обработката, се извършва записване във файл *output.bmp*.

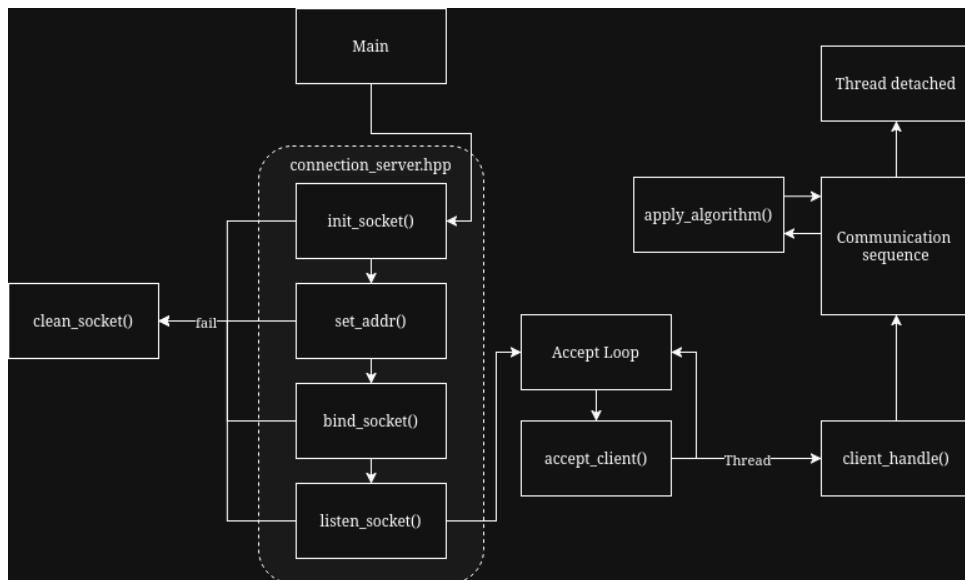


Сървър

Сървърът започва, като подготвя комуникационния канал. След като конфигурира адресите и нужните настройки, той започва цикъл, чакайки нови връзки.

Ако нов клиент се свърже, той се поема от нишка, която се създава и отделя. Отделянето е нужно, за да не чака основният процес. Вместо това той продължава да поема нови връзки.

Когато се предаде снимката от клиента, се прилага избраният алгоритъм и резултатът се връща на клиента.



Алгоритми

Съществуват няколко вградени алгоритъма за обработка на изображения. В скобите са записани номерата за извикването им:

- **Обръщане на цветовете (1.)** - инвертира стойностите на цветните канали на всеки пиксел.
- **Сива скала (2.)** - преобразува изображението в сива скала, използвайки формулата
$$Gray = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B.$$
Получената стойност се записва еднакво в каналите R, G и B на пиксела.
- **Черно-бяло (4.)** - изчислява се яркостта на пиксела; ако стойността надвишава 127, пикселят става бял (255), а в противен случай – черен (0).
- **Засичане на ръбове (Сobelов филтър, 8.)** - използва две 3×3 матрици за хоризонтално и вертикално откриване на ръбове. Изчислява се силата на разликата между съседни пиксели, като по-ярките стойности означават по-силно изразени ръбове.
- **Размазване (Гаусов филтър, 16. и 32. за многонишкова)** - прилага се двумерен Гаусов филтър, базиран на Гаусова функция. Алгоритъмът поддържа многонишкова обработка.
- **Габоров филтър (64. и 128. за многонишкова)** - комбинира Гаусово затихване и косинусоидална функция за извлечане на текстурни и ориентационни характеристики. Филтърът подчертава хоризонтални линии и текстури. Алгоритъмът поддържа многонишкова обработка.

Мрежова комуникация

Комуникацията между сървъра и клиента се случва едва когато снимката е заредена и готова за изпращане.

Установява се комуникация за потвърждение и искане, като се използва малък 3-байтов пакет.

Първият байт се използва за основна комуникация. Има три дефинирани стойности: 0x00 за успех, 0x01 за грешка и 0x19 за искане на модификация от клиента към сървъра.

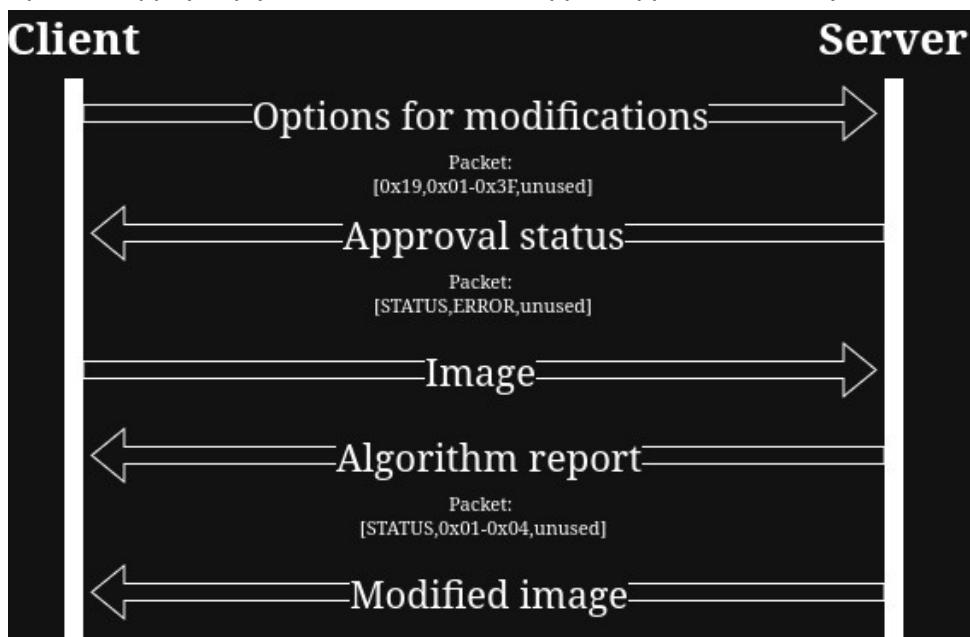
Вторият байт при пакет с искане (0x19) се използва за изпращане на вида модификация - от 1 до 255 (0x01 до 0xFF). Модификациите могат да се наслагват като битови маски. Например 5 ще извика обръщане на цветовете (1) и черно-бяло (4).

При пакет с грешка (0x01) вторият байт се използва за допълнителна информация. При грешка 0x01 файлът не е в BMP формат. При 0x02 файлът е версия 1, която не се поддържа. При 0x04 файлът е BMP формат, но е значително по-малък, отколкото трябва да бъде.

Третият байт е оставен празен и може да се използва за бъдещо развитие.

Процесът на комуникация е следният:

1. Клиентът пита за модификация [0x19, 0x01-0x3F, -]
2. Сървърът връща отговор:
 1. При валидна модификация [0x00, -, -]
 2. При невалидна модификация [0x01, -, -]
3. Клиентът изпраща снимката
4. Сървърът връща доклад:
 1. При валиден файл [0x00, -, -]
 2. При невалиден файл [0x01, 0x01-0x04, -]
5. Връща се модифицираната снимка, ако доклада е с отговор 0x00



Обработка на грешки и сигурност

По време на четене на файлове от клиента се осигурява сигурност, че файлът е в подходящия формат, чрез четене на името на файла и след това четене на header-а на файла. Ако нещо не отговаря на BMP стандарта, се изхвърля грешка и се почиства паметта.

Същият процес е подобен и за сървъра. Ако приетите данни са грешни, се изпраща грешка и се почиства заделената памет. Това може да бъде лош формат, невалиден header или недостатъчен брой байтове.

При създаване на комуникация също се проверява за грешка и при нужда се извършва почистване. Възможна грешка при комуникацията е изпращането на недостатъчен брой байтове.

Скорост на работа

Всички тестове са правени при локална комуникация, което елиминира забавянето в мрежата. Компютърът поддържа 8 нишки на съвременна архитектура.

При тестване на по-простите алгоритми (Обръщане на цвета, Сива скала, Черно-бяло, Засичане на ръбове) средно се постига време за обработка от 100 до 500 милисекунди.

При по-сложните алгоритми (Размазване и Габоров филтър) времето за обработка е между 6 и 14 секунди.

При включване на няколко нишки при сложните алгоритми успешно е намалено времето до между 2 и 4 секунди, което е приблизително 3 пъти по-бързо.

Пример

Клиента предоставя доста функции, но за този пример са важни две load и send.

„load“ използва относителен път за отваряне на изображения (например load ../../files/big_input.bmp).

„send“ изпраща снимката на сървъра. Може да се напише send <число> за директно изпращане или просто send, за да се покаже меню с алгоритми, след което се пише числото на избраните алгоритми и се изпраща. В този пример това са 17 или 33 (ако желаем многонишково).

За целта е избрана малка снимка win_v5.bmp около 1.6 Mb от папката /files.

Тази снимка се зарежда от клиента и се подава на сървъра, при което той връща отговор с дадените филтри. Тук са показани филтри за обрънат цвят и размазване

От ляво на дясно са:

оригиналната -> обрънат цвета -> размазване -> обрънат цвят+ размазване.



Ограничения и бъдещо развитие

Възможно е реализирането на повече алгоритми и лесното им добавяне чрез промяна на ограничението, така че да се приемат повече комбинации.

Максималният размер на снимките е около 25 MB, което се равнява на изображение с размер 3000×4000 пиксела. Това може да бъде променено в BMP_File.hpp.

Филтрите могат да се комбинират, но редът им е фиксиран и не може да се променя. Например филтърът за сив цвят не може да бъде след засичането на ръбове и т.н.

Въпреки че е мултиклиентски сървър, не се използват сложни структури за следене на клиентите. Това прави сървъра уязвим на претоварване поради много клиенти.

Краен резултат

Разработената програма позволява зареждане на изображения в BMP формат, тяхното изпращане към сървър и последваща обработка с избрани алгоритми. Реализиран е механизъм за откриване и обработка на грешки, който гарантира коректна работа при невалидни файлове или проблеми в комуникацията. Налични са няколко алгоритъма за обработка на изображения, които могат да се комбинират по предварително дефиниран ред. Архитектурата на системата позволява лесно разширяване с нови алгоритми и функционалности в бъдеще.