Курсова работа

по

Системно програмиране

на тема

„ftp server“

Изготвил: ........................................... /Николай Чобанов/

Студент в ІІІ-ти курс на ТУ-София специалност КСИ, група 44, факултетен номер 121214189

Проверил: ..........................................

/ас. Десислав Андреев/

Дата на предаване: 1.06.2017г.

***Съдържание:***

1. Анализ на изготвеното приложение.

* Как разбирам условието.

1. Функционално описание

* Функционално описание на сървърната част.
* Функционално описание на клиентската част.

1. Sequence diagram.
2. Изпълнение на функционалностите.

* Сървър:
* Клиент:

1. Криптация и Декриптация.
2. Тестове:

* Стартиране на клиента и сървъра.
* Тест с правилна парола, връзката е установена, тест на функция 4 ls
* Тест на 5 cd.
* Тестване на функция 1 get.
* Тест на функция 3 pwd
* Тест на функция 2 put
* Тест за нормално напускане на сървъра.

1. Използвана литература.
2. ***Анализ на изготвеното приложение.***

(Задача с курсов проект) Да се направи приложение, което реализира FTP сървър. Нужно е да се реализират поне най-основните команди, за да е възможно да се автентикира с име и парола, да се прелистват файлове, да се сменят директории, да се изтеглят файлове и да се качват файлове. За повече информация относно командите на протокола вижте: http://www.cs.colostate.edu/helpdocs/ftp.html Ако нямате акаунт на действащ сървър, можете да си инсталирате FileZilla Server на локалния компютър и да го достъпвате през localhost от тук: http://filezilla-project.org/ При работа в екип и студент с курсова работа: Да се реализира поддръжка на криптиране на информацията (FTPS, SFTP и др.) – нека се поддържа само стандартния, некриптиран протокол.

* **Как разбирам условието?**

Моята идея е да направим две приложение. Едното да реализира основните функции на ftp сървъра а другото да играе ролята на клиент.

За сървъра – има списък с пароли съхраняващ се във файл, сървъра трябва да може да го достъпва и да изтегля информацията от него. След установяване на връзка с клиента да го оторизира, в случай че полученото потребителско име/парола е грешно да прекъсва връзката. Паролата да се получава криптирана по наш алгоритъм и се декриптира. Паролите във файла да се съхраняват в декриптиран вид. Командите и трансфера на файлове да не се криптира.

Клиентска част – клиента установява връзка със сървъра, чака парола от стандартния вход, като се въведе парола я криптира и я изпраща и чака отговор за оторизация. При получен отговор за грешна парола процеса се терминира, при получен отговор за правилна парола очаква команди въведени от клавиатурата и ги изпраща на сървъра, очаквайки след това съответните отговори.

1. ***Функционално описание***

* **Функционално описание на сървърната част.**

Установява връзка с клиента, проверява дали паролата е правилна и изпълнява следните команди:

1. get – копиране на файл от сървъра на локалната машина

2. put – копиране на файл от локалната машина на сървъра

3. pwd – намиране на пътя на текущата директория на сървъра

4. ls – извеждане на имената на файловете от текущата директория на сървъра

5. cd – промяна на директория на сървъра

6. quit – изход.

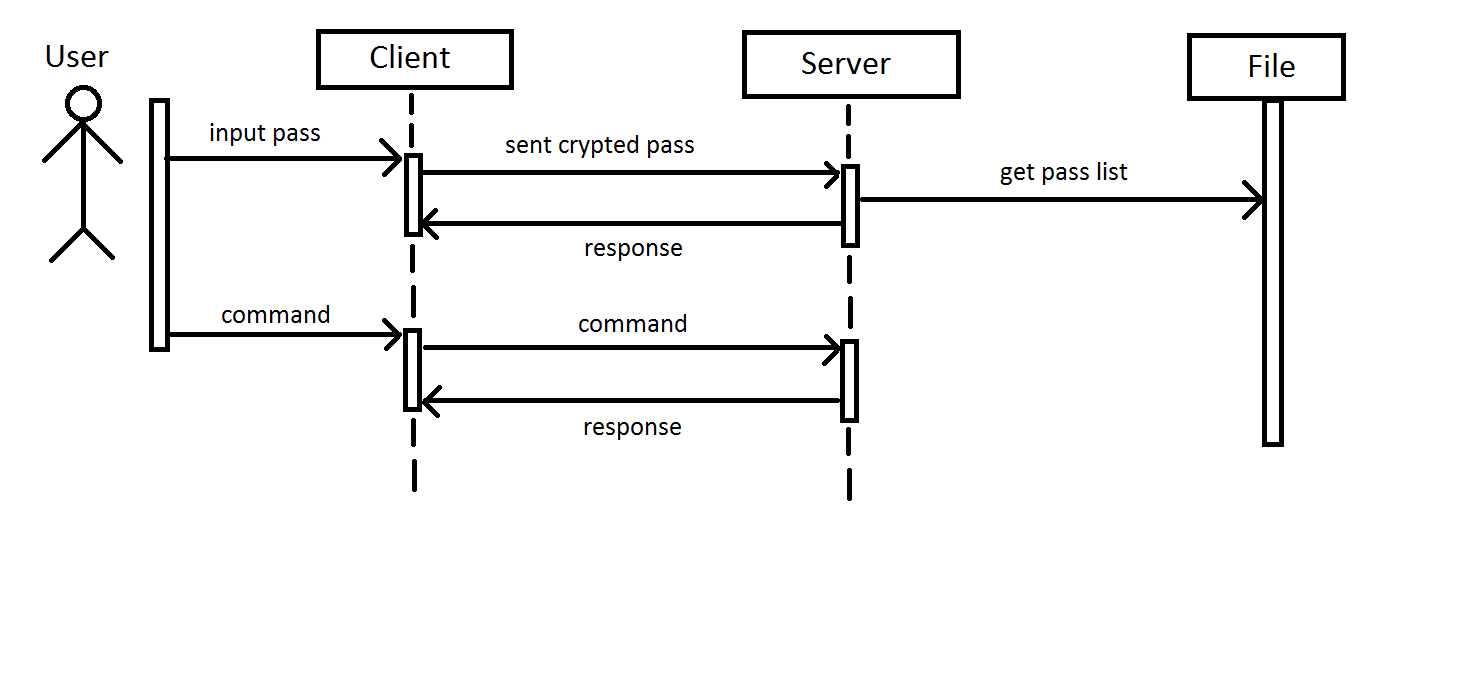
При грешна парола връзката със сървъра се прекъсва

Криптиращите функции са имплементирани в отделен файл и са описани в точка 4 от документацията.

* **Функционално описание на клиентската част.**

Установява връзка със сървъра, криптира паролата и я изпраща, очаква отговор дали е приета паролата. Изпраща команди и очаква съответните отговори.

1. ***Sequence diagram.***



1. ***Изпълнение на функционалностите.***

* **Сървър:**

Създаване на сокет:

sock1 = socket(AF\_INET, SOCK\_STREAM, 0);

Сокета започва да „слуша“:

k = listen(sock1,1);

Получава парола от клиента.

recv(sock2, buf, 100, 0);

Декриптира получената парола.

decript(buf);

Проверява дали паролата е правилна

if(checkPass(pass)!=1)

Получава команта от клиента и я записва в „buf”:

recv(sock2, buf, 100, 0);

При избрана “ls“ команда, изпълнява командата на сървъра и информацията я запетява във файл, който после го изпраща на клиента.

system("ls > temporary.txt");

sendfile(sock2,filehandle,NULL,size);

Взима информацията за файла:

stat(filename, &obj);

* **Клиент:**

Инициализира номера на порта и адреса:

server.sin\_family = AF\_INET;

server.sin\_port = atoi(argv[1]);

server.sin\_addr.s\_addr = 0;

Свързва се със сървъра:

k = connect(sock,(struct sockaddr\*)&server, sizeof(server));

Приема парола от клавиатурата:

scanf("%s", buf);

Криптира я.

cript(buf);

Изпраща паролата:

send(sock, buf, 100, 0);

Приема от сървъра дали паролата е вярна или не:

recv(sock, buf, 100, 0);

Инициализира „buf“ според дадената команда:

strcpy(buf, "get ");

Добавя “filename“ към “buf”:

strcat(buf, filename);

Записва получената информация от сървъра върху дадения файл:

write(filehandle, f, size, 0);

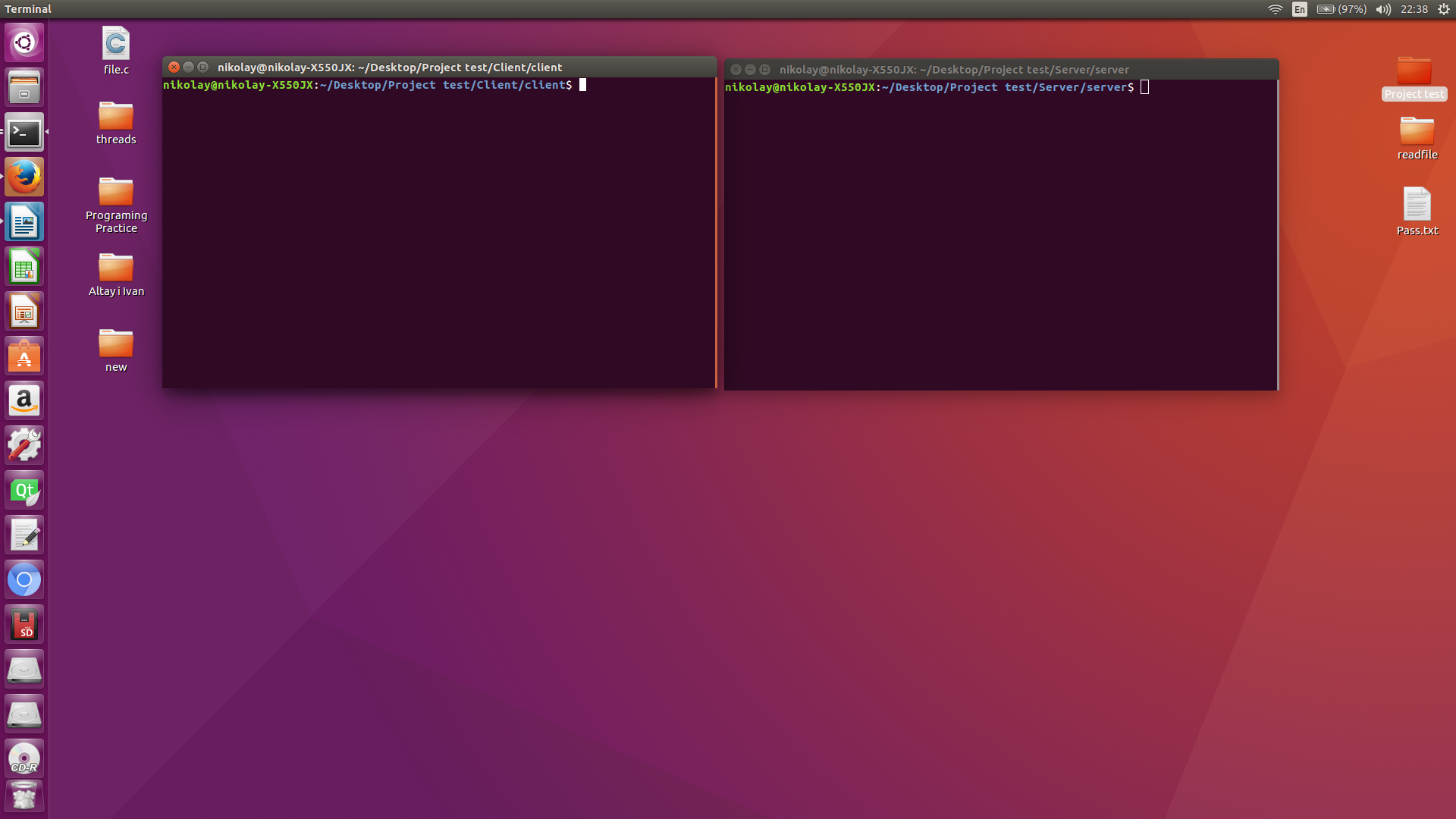
1. ***Криптация и Декриптация.***

Криптацията е доста проста и е изцяло имплементирана от нас, приема се char масив и към кода от ASCII таблицата на всяка буква се добавя поредния и номер на символът в стринга.

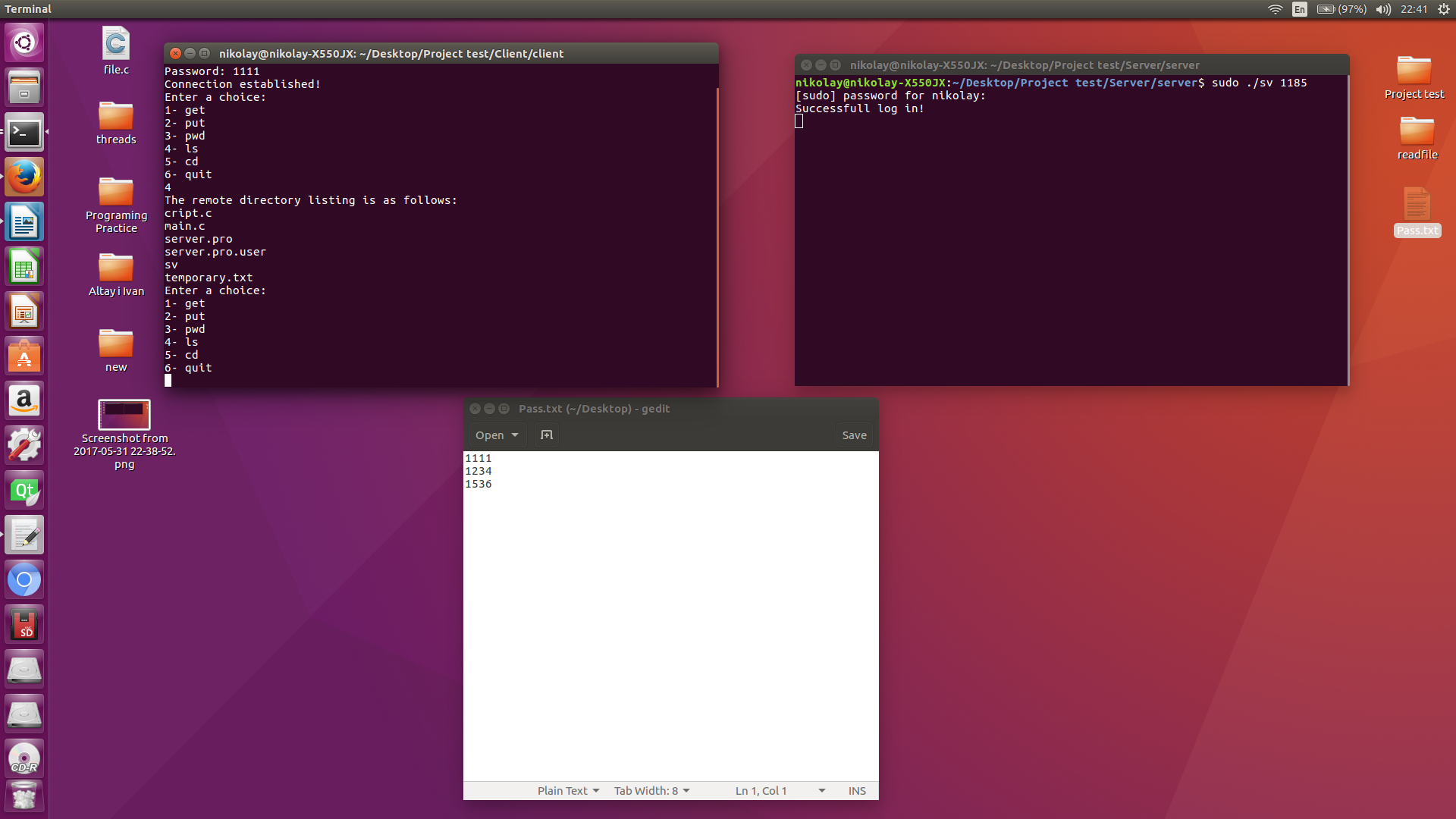
Декриптацията се осъществява по обратния ред

1. ***Тестове:***

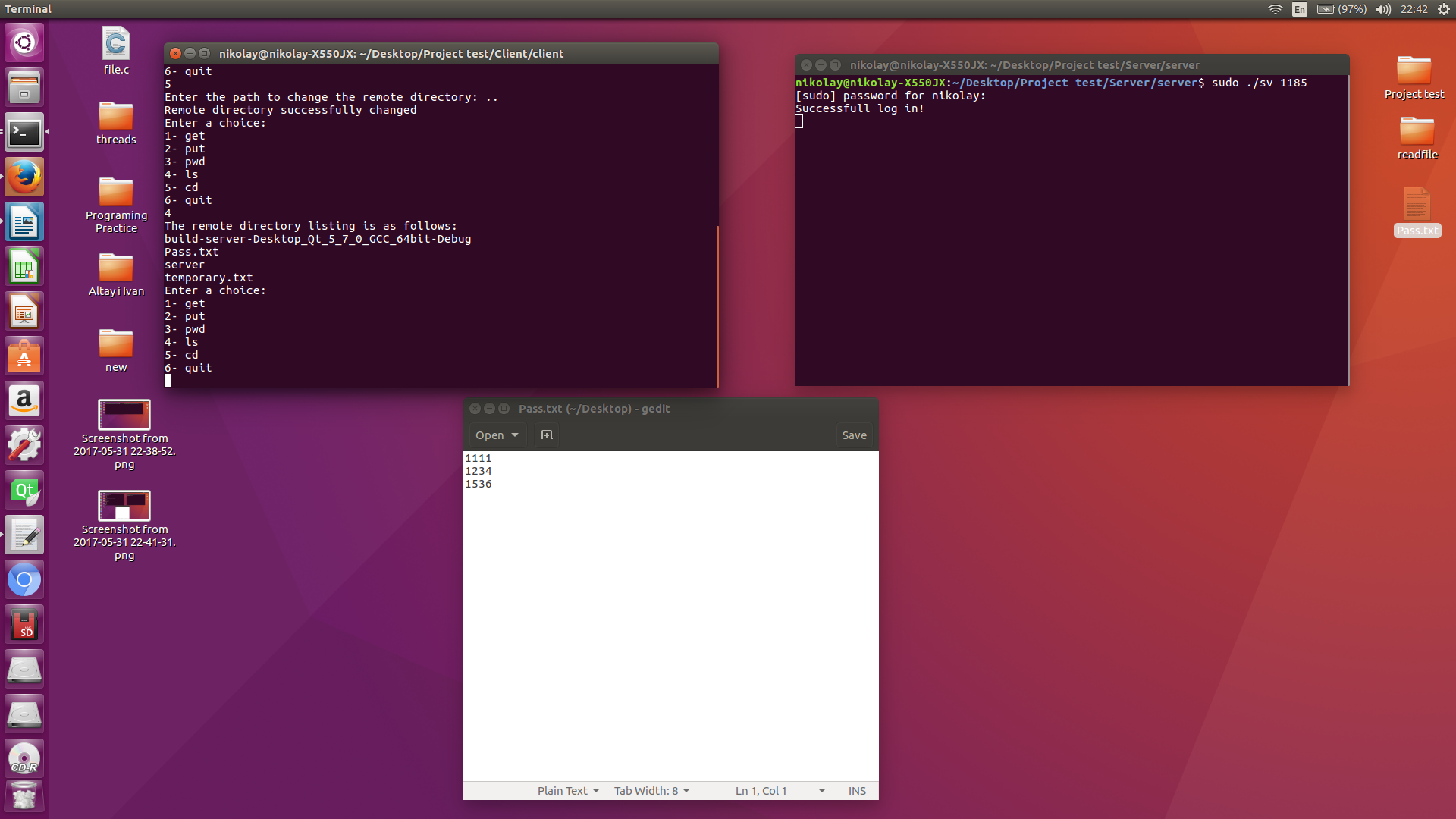
* **Стартиране на клиента и сървъра.**



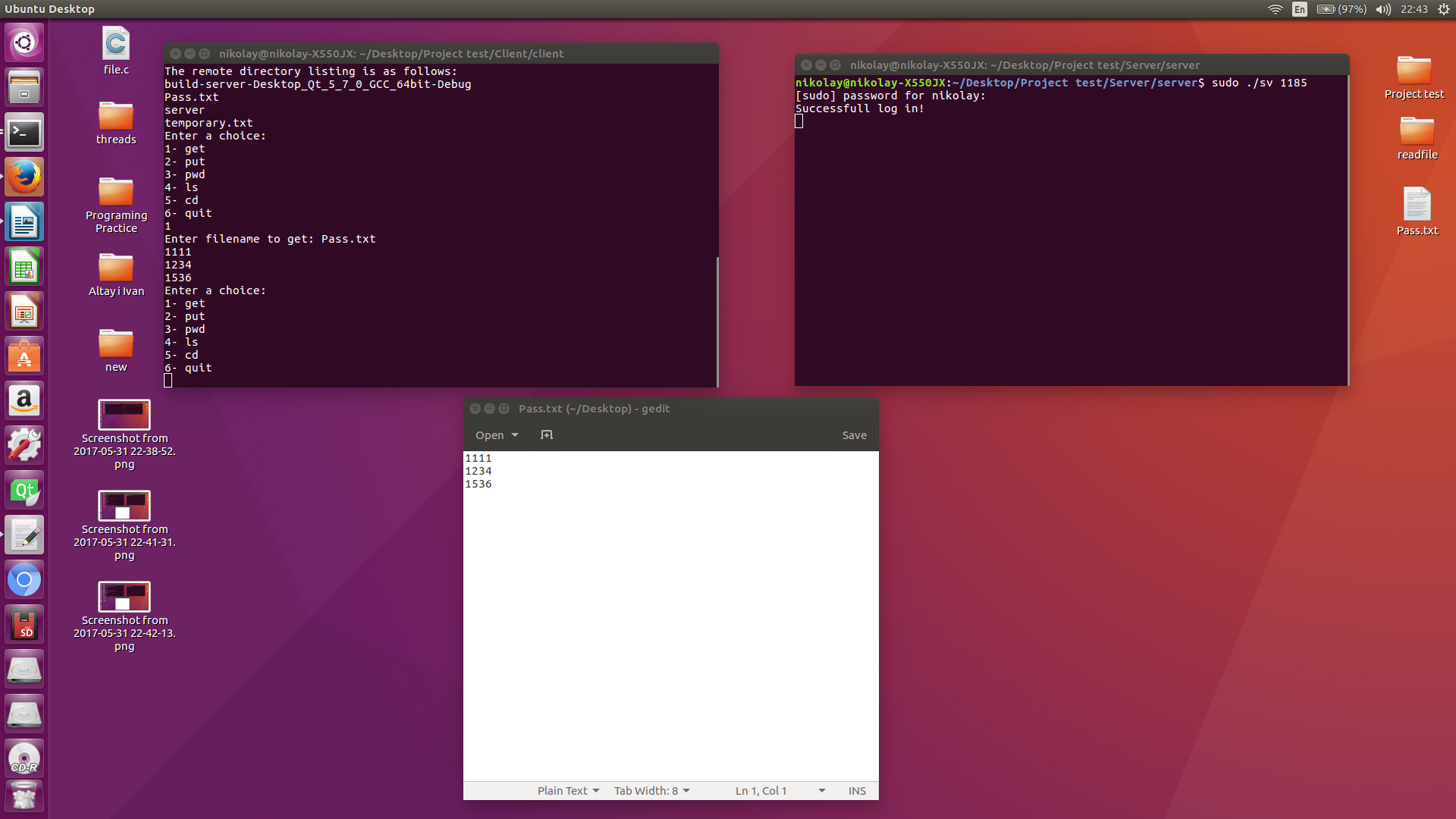
* **Тест с правилна парола, връзката е установена, тест на функция 4 ls**



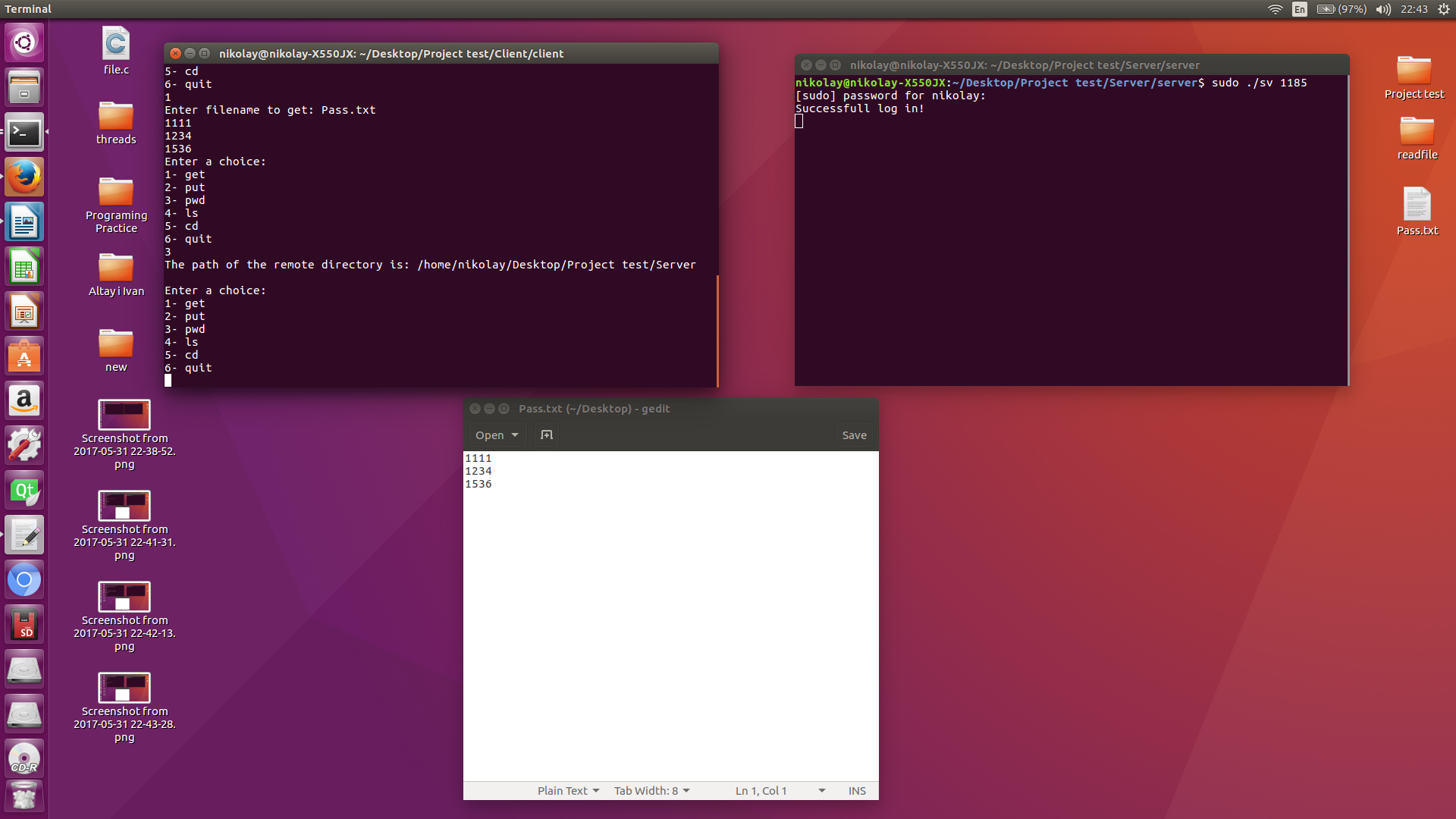
* **Тест на 5 cd.**



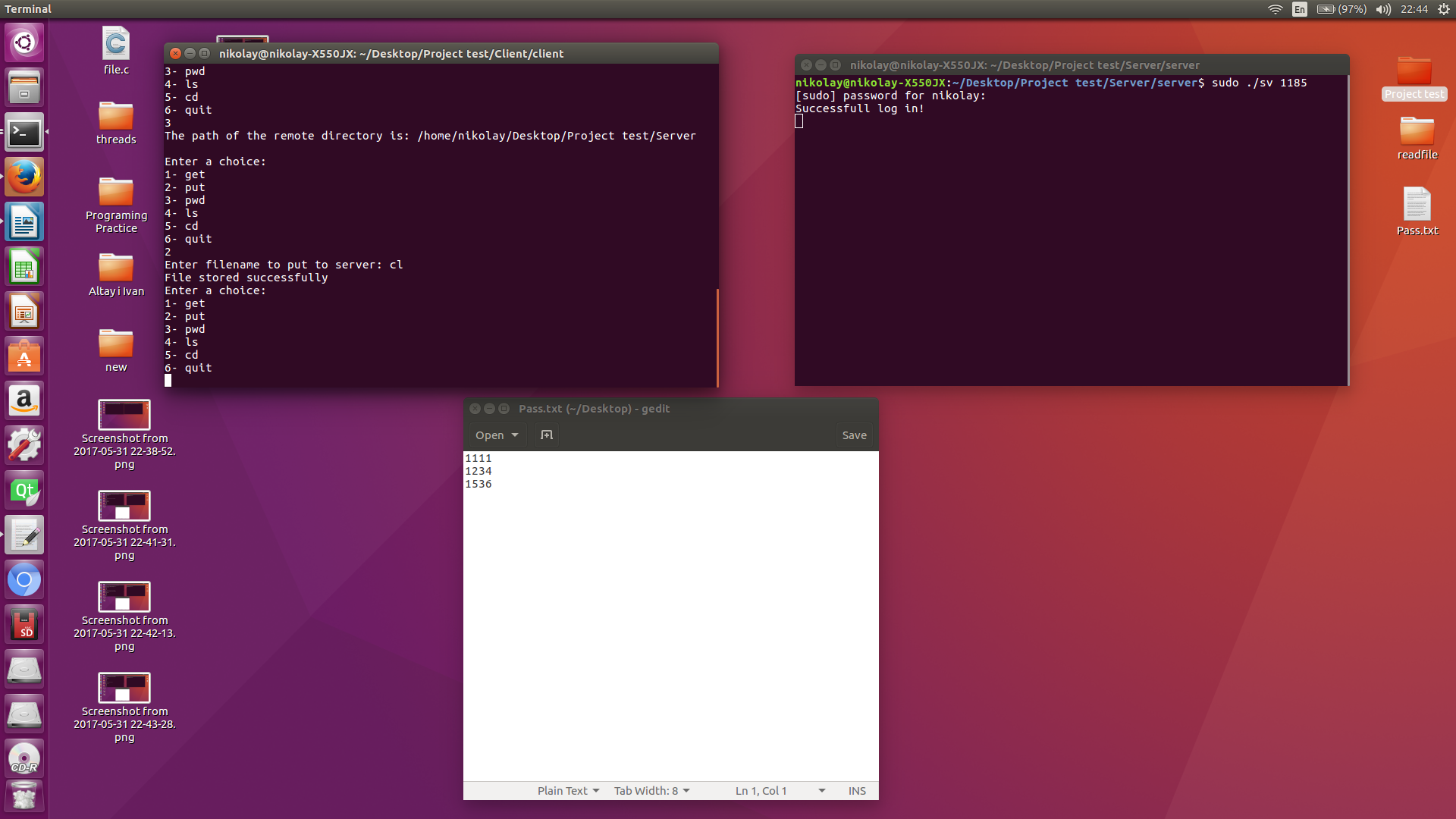
* **Тестване на функция 1 get.**



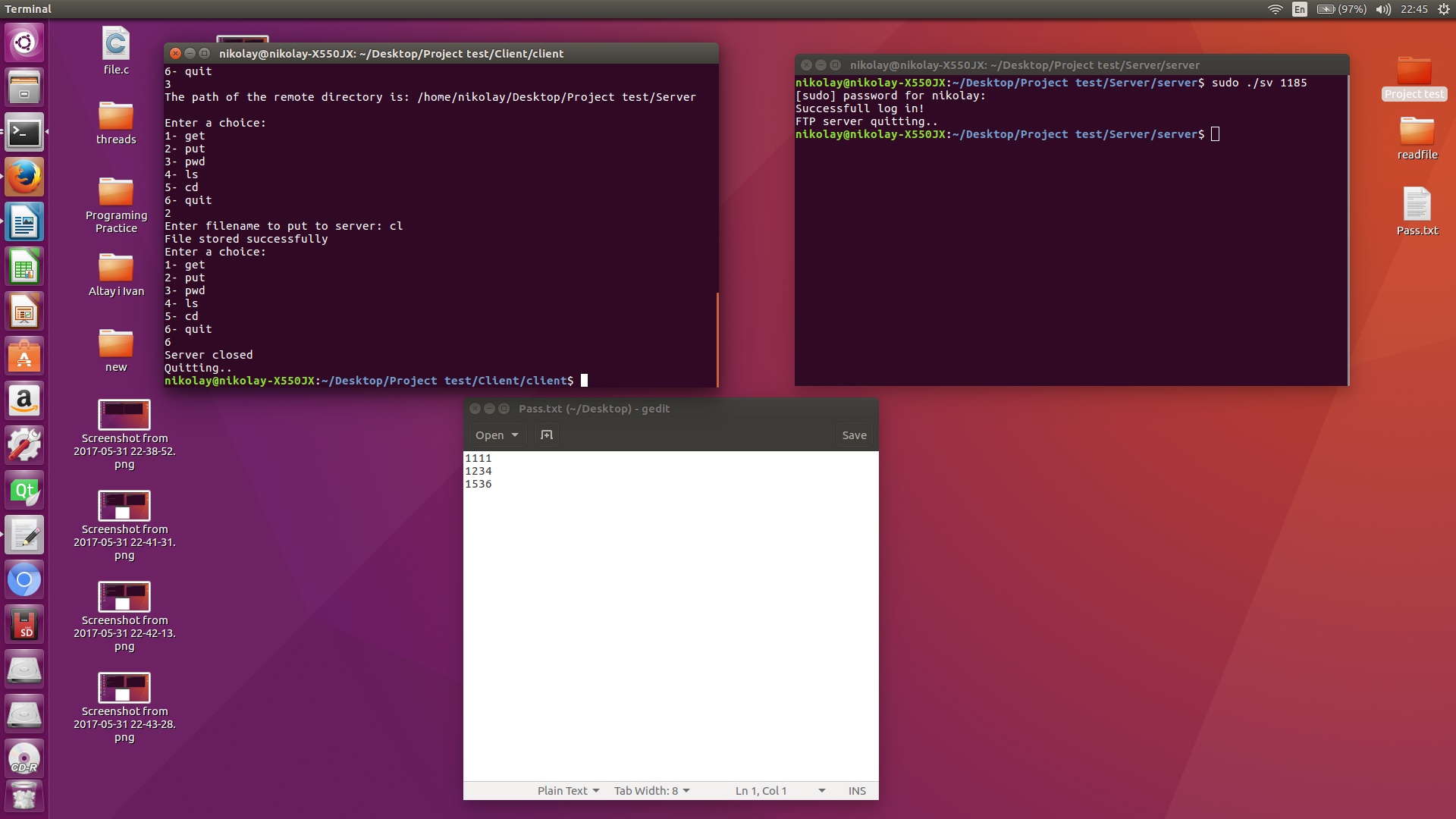
* **Тест на функция 3 pwd**



* **Тест на функция 2 put**

****

* **Тест за нормално напускане на сървъра.**



1. ***Използвана литература.***

* [**http://www.cs.colostate.edu/helpdocs/ftp.html**](http://www.cs.colostate.edu/helpdocs/ftp.html)
* [**https://en.wikipedia.org/wiki/File\_Transfer\_Protocol**](https://en.wikipedia.org/wiki/File_Transfer_Protocol)