Министерство высшего и профессионального образования РФ

Южно Уральский государственный университет

Кафедра «Системного программирования»

Лабораторная работа №1

Простое сетевое приложение

Выполнил:

студенты группы ВМИ-356

Лукичева Е.М. \_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

Громов И.А. \_\_\_\_\_\_\_\_\_\_\_

Челябинск

2013

## Цель работы

1. Ознакомиться с языком программирования Python
2. Ознакомиться с работой системы контроля версий Git
3. Написать сетевое приложение и сохранить результат работ в репозитории Git

## Техническое задание

Написать эхо-сервер. Сервер должен слушать указанный TCP порт, транслировать данные из входящего соединения в исходящее.

## Ход работы

### Как я создавал репозиторий

Мы создали репозиторий на сайте, а затем скопировали его на локальный компьютер.

Сперва мы создали аккаунт на сайте GitHub.com. Для создания нового репозитория перешли в профиле на вкладку Repositories и нажали кнопку New. Указываем имя нового репозитория (так же можем задать описание, игнорируемые файлы, указать необходимость создания файла README), и он будет готов.

Для работы с репозиториями на локальном компьютере мы использовали программу GitHub for Windows. Входим в программу, используя логин и пароль, которые мы использовали для создания аккаунта на GitHub. Если мы откроем меню local->repositiories, программа отобразит нам все репозитории, привязанные к пользователю на сайте GitHub. Для того чтобы создать локальную копию какого-либо репозитория у себя на локальном компьютере, нужно нажать на кнопку с пометкой “CLONE”.

### Работа с сокетами в python

Для изучения работы сокетов в python, мы написали две программы: клиентскую и серверную часть. Клиент посылает серверу строку «PALINDROM», а затем ждет возвратного сообщения от сервера. Сервер принимает входящие сообщения, инвертирует порядок символов в нем и отправляет обратно всем клиентам.

Таблица client.py

|  |
| --- |
| **import** socket  HOST **=** "" *# remote computer (localhost)*  PORT **=** 33333 *# port on the remote computer*  sock **=** socket**.**socket(socket**.**AF\_INET, socket**.**SOCK\_STREAM)  sock**.**connect((HOST, PORT))  sock**.**send("PALINDROM")  result **=** sock**.**recv(1024)  sock**.**close()  **print** "Received: ", result |

Таблица server.py

|  |
| --- |
| **import** socket**,** string  **def** **do\_something**(x):      lst **=** map(None, x);      lst**.**reverse();  **return** string**.**join(lst, "")  HOST **=** "" *# localhost*  PORT **=** 33333  srv **=** socket**.**socket(socket**.**AF\_INET, socket**.**SOCK\_STREAM)  srv**.**bind((HOST, PORT))  **while** 1:  **print** "I'm listening port 33333"      srv**.**listen(1)      sock, addr **=** srv**.**accept()  **while** 1:          pal **=** sock**.**recv(1024)  **if** **not** pal:  **break**  **print** ("Received from %s:%s:" **%** addr, pal)          lap **=** do\_something(pal)  **print** ("Sent %s:%s:" **%** addr, lap)          sock**.**sendall(lap)      sock**.**close() |

Программы были написаны в среде разработки JetBrains PyCharm 2.7.

### Сохранение изменений в репозиторий

Для сохранения изменений в репозитории зайдем в командную строку из папки, в которой находится содержимое репозитория на локальном компьютере.

Для начала нужно указать, какие файлы следует учитывать при следующем commit'e. Это делается с помощью следующей команды:

git add <Имя файла>

Можно добавлять не только одиночные файлы, но и целые папки.

Затем делаем commit изменений. Для этого вводим следующую команду:

git commit –m “Мы сделали эхо сервер. Теперь делаем коммит.”

Для того чтобы наши изменения отразились в репозитории, необходимо сделать push. Для этого в командной строке вводится следующая команда:

git push

## Результат

В ходе лабораторной работы мы:

1. Познакомились с языком программирования Python.
2. Научились создавать репозитории, делать commit'ы в репозитории, обновлять изменения файлов репозитория.