System Programming
6th Laboratory (8th and 13th April 2016)

In this laboratory the students will implement a system similar to the one to be implemented in the project.
The system to be developed in the project will be composed of serverl multi-threaded servers.
the system to implement in this laboratory will ehxibit the same behaviour (with respect to data) but will serv a single glient at a time.

1 Key-value store

The objective of this Laboratory is to develop a simple key-value store system.
The system will be composes of a programming API, a library executed in the client applications and a single server.

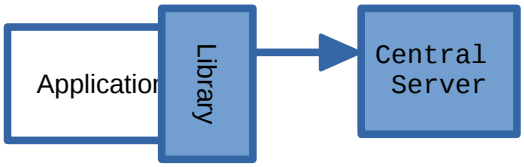
1.1 Definition

From Wikipedia:

> A key-value store, or key-value database, is a data storage paradigm designed for storing, retrieving, and managing associative arrays, a data structure more commonly known today as a dictionary or hash. Dictionaries contain a collection of objects, or records, which in turn have many different fields within them, each containing data. These records are stored and retrieved using a key that uniquely identifies the record, and is used to quickly find the data within the database.

1.2 Architecture

The overall architecture of the system is presented in the next figure (left centralized version, right distributed version):



Only the components in blue are developed in the context of this project. A testing application will be provided.

The library is composed of a set of data-types and functions that external programmers can use to interact with the database. The interface (API) of this library is described in this document. The functions of this library will be implemeneted by the students and will allow the access of data stored the server.

1.3 API

The system should provide a programming API that allows the development of programs that use the key-value store to store values.

The required functions to implement are:

int kv_connect(char * kv_server_ip, int kv_server_port);

This function establishes connection with a Key-value store. The pair (**kv_server_ip** and **kv_server_port**) corresponds to the address of the Controller

This function return **-1** in case of error and a positive integer representing the contacted key-value store in case of success. The returned integer (key-value store descriptor) should be used in the following calls.

void kv_close(int kv_descriptor)

This function receives a previously opened key-value store (**kv_descriptor**)and closes its connection.

int kv_write(int kv_descriptor, uint_32 key, char * value, int value_length)

This function contacts the key-value store represented by **kv_descriptor** and stores the pair (**key**, **value**). The **value** is an array of bytes with length of **vaue_length**. The system returns 0 in case of succes and -1 in case of error.

int kv_read(int kv_descriptor, uint_32 key, char * value, int value_length)

This function contacts the key-value store and retrieves the value corresponding to key. The retrieved value has maximum length of **value_length**. And is stored in the array pointed by **value** .

int kv_delete(int kv_descriptor, uint_32 key)

This function contacts the key-value store to delet the value corresponding to key. From this moment on any kv_read to the suplied key will return error.

These functions should be implemented as a library (psiskv_lib.c and psiskv.h files) that other programmers can add to their own code.

2 Laboratory implemenation

The students should develop the library (psiskv_lib.c and psiskv.h files) and the server.

The server will listen to the port 9999 or first available port and will handle a single client a a time:

> The server receives an accept (from the **kv_connect**) will read all the requests from the client (**kv_write** and **kv_read**). Only after the **kv_close** on the client the server should do a new accept.

Students should define the structure of the messages exchanged between the client and the server:

- From client to the server

    - **kv_write** - key, value, length of the value
    - **kv_read** - key-value
- From the server to the client
    - **kv_write** - notification of success
    - **kv_read** - value, length of value

The pairs (key, value) should be stored in a suitable data-structure. The simplest being a linked list.

The communication should be done using SOCK_STREAM AF_INET sockets.

3 Project

The code of the Library (on the clients), the data-structure (on the server), and the message structures will remain the same on the final project.