БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе

Тема: «Методы взаимодействия в звене сети передачи данных»

Выполнил:

студент группы 950503 Пастернак А.С.

Проверил:

ассистент Марцинкевич В.А.

Минск

2021

Задание к лабораторной работе:

1. Написать программу передачи данных по протоколу TCP/IP.

   Доп. условие: ожидать подтверждения успешного принятия пакета.

Листинг программы:

```
class Socket {
protected:
   int sDescriptor;

   int domain;
   int type;
   int protocol;
   int port;

   struct sockaddr_in sAddr;

public:
   constexpr Socket() : domain{AF_INET}, protocol{SOCK_STREAM}, type{0}, port(8080),
sDescriptor{-1} {
      memset(&sAddr, 0, sizeof(sockaddr));
   }
   constexpr Socket(const int domain, const int protocol,
                    const int type, const unsigned int port) : domain{domain}, protocol{protocol},
type{type}, port(port), sDescriptor{-1} {
      memset(&sAddr, 0, sizeof(sockaddr));
   }

   virtual ~Socket() {
      stop();
   }

   void change_domain(const int domain) noexcept;
   void change_type(const int type) noexcept;
   void change_protocol(const int protocol) noexcept;

   virtual void start();
   virtual void stop();

   virtual bool is_open();

protected:
   void create();
   void init_server_sockaddr() noexcept;
   void init_client_sockaddr() noexcept;
   void bind_name();
};
```

```
class Station : public Socket {
private:
    int cDescriptor;
    unsigned int queue;

public:
    constexpr Station() : cDescriptor(-1), queue(20) {}
    constexpr Station(const unsigned int queue) : cDescriptor(-1), queue(queue) {
    }

    ~Station() {
        stop();
    }

    void change_queue(const unsigned int queue) noexcept;

    void start() override;
    void stop() override;

    bool is_open() override;

    void send_pack(Package &P);
    Package& get_pack();

    void set_queue_connect(const unsigned int &queue);
    void accept_connection();
private:

    unsigned int get_status();
    void send_status(Status &S);
};

class Client : public Socket {
private:
    IPv4 ip;
public:
    constexpr Client() : ip{} {
        memset(&sAddr, 0, sizeof(sAddr));
    }
    constexpr Client(const IPv4 &ip) : ip(ip) {
        memset(&sAddr, 0, sizeof(sAddr));
    }

    void start() override;
    void convert_ip();

    void send_pack(const Package &pack);
    Package& get_pack();

    void set_ip(const IPv4 &ip);
private:
```

```
    void connect_to_server();

    unsigned int get_status();
    void send_status(Status &S);
};

class IPv4 {
private:
    std::array<unsigned char, 4> data;
public:
    constexpr IPv4() : data{ {0} } {}
    constexpr IPv4(unsigned char const a, unsigned char const b,
                   unsigned char const c, unsigned char const d)
                       : data{{a, b, c, d}} {}

    constexpr IPv4(IPv4 const &other) noexcept : data(other.data) {}

    IPv4& operator=(IPv4 const &other) noexcept;

    std::string to_string() const noexcept;
    unsigned long to_unlong() const noexcept;

    void from_unlong(const unsigned long data) noexcept;

    friend std::ostream& operator<<(std::ostream &os, const IPv4 &a);
    friend std::istream& operator>>(std::istream &is, IPv4 &a);
};


const size_t MAX_SIZE_PACK = (sizeof(Pack) + 100) * 2;

class Package {
private:
    struct Pack value;
    IPv4 sender;
    IPv4 recipient;

public:
    constexpr Package() : sender{}, recipient{}, value{} {}
    constexpr Package(const IPv4 &sender, const IPv4 &recipient) : sender{sender},
recipient{recipient}, value{} {}

    void start() noexcept;

    char* get_data() noexcept;

    void change_sender(const IPv4 &sender) noexcept;
    void change_recipiend(const IPv4 &recipiend) noexcept;

    template <typename Type>
    void change_data(Type &data) {
```

```
        this->value.fother = typeid(data).hash_code();

        memset(this->value.data, '\0', MAX_SIZE_PACK_DATA);
        memcpy(this->value.data, data, sizeof(data) * sizeof(*data));

        this->value.sizeData = sizeof(data);
    }

    template<class Archive>
    void save(Archive &ar, const unsigned int version) const {
        ar & this->value;
    }

    template<class Archive>
    void load(Archive &ar, const unsigned int version) {
        ar & this->value;
    }

    BOOST_SERIALIZATION_SPLIT_MEMBER()

    friend std::ostream& operator<<(std::ostream &out, Package &P);
//    friend std::istream& operator<<(std::istream &in, Package &P);
private:
    void set_flag() noexcept;
    void set_sender() noexcept;
    void set_recipiend() noexcept;
    void set_other_flag() noexcept;
    void set_size_pack() noexcept;

    friend class boost::serialization::access;

private:
    const unsigned long fstart = 02001;
};
```

Заключение:

В данной лабораторной работе разработан модуль передачи данных по протоколу TCP/IP.