

# **Системное программирование**

## **Л.р.7. Элементы сетевого программирования**

### **Цель:**

Основы построения и функционирования сетей, стеки протоколов, программные интерфейсы. Организация взаимодействия между процессами в сетях.

### **Теоретическая и методическая часть**

Иерархическая модель взаимодействия открытых систем, уровни, протоколы и интерфейсы. Стеки протоколов OSI, TCP/IP и др.

Протокол IP. Идентификация абонентов в IP-сетях. Протоколы транспортного уровня TCP и UDP.

Программный интерфейс сокетов. Типы, структуры данных, функции API.

Порядок организации взаимодействия посредством сокетов (поточковых и датаграмных). Способы организации серверов.

Функции (вызовы) `socket()`, `bind()`, `connect()`, `listen()`, `send()`, `recv()`, `sendto()`, `recvfrom()` и др.

### **Практическая часть**

#### **Общая постановка задачи:**

Программа или несколько программ, демонстрирующие взаимодействие между собой или с иным (внешним) ПО посредством сетевых протоколов и программного интерфейса сокетов. Реализация известных сетевых протоколов прикладного уровня или разработка собственных.

Специальных требований к приложениям не предъявляется; в частности, во многих случаях они могут быть не обязательно оконными, но также и консольными.

#### **Варианты заданий:**

- интерактивное взаимодействие пользователей («чат»)
- удаленный командный интерпретатор
- сервер/клиент службы прикладного уровня
- сервер/клиент стандартной сетевой службы
- аналог сетевой утилиты
- ...

## **1 Интерактивное взаимодействие пользователей («чат»)**

Упрощенный чат для нескольких пользователей (в целом подобно заданию из темы «Обмен данными») с использованием сетевых сокетов.

Транспортные протоколы: TCP или UDP.

Архитектура: централизованная (выделенный процесс-сервер и процессы-клиенты) или децентрализованная (процессы-клиенты с «серверными» функциями).

Сервер: создание сокета для приема соединений или отдельных сообщений; прием и временное хранение сообщений; передача сообщений адресно одному или нескольким клиентам; поддержание списка актуальных клиентов.

Клиент: обнаружение сервера и соединение с ним; ввод пользовательских сообщений и передача их серверу либо напрямую соответствующему клиенту; прием и отображение сообщений.

Потребуется разработка структуры передаваемых сообщений (как минимум, адрес и тело сообщения) и порядка обмена (протокола прикладного уровня).

Опционально: в децентрализованной системе маршрутизация сообщений (возможность трансляции сообщений по цепочке участников).

## **2 Удаленный командный интерпретатор**

Разработка и реализация командного интерпретатора («оболочки», shell) с «дистанционным управлением» (включая протокол прикладного уровня для взаимодействия с ним). Язык интерпретатора редуцированный – для целей демонстрации.

Агент (сервер): установка на заданный порт; прием запросов (команд для исполнения); исполнение поступивших команд в локальной системе; возврат результата (опционально, если предусмотрено для конкретной команды).

Менеджер (клиент): соединение с агентом; ввод (чтение из сценария) команд; передача команд агенту; контроль результативности.

Набор команд может быть как собственный (синтезированный), так и совпадающий с «родными» командами в среде локальной системы. Во втором случае для их выполнения можно воспользоваться вызовом `system()` (в Win поддерживается, но с особенностями).

### **3 Сервер/клиент службы прикладного уровня**

Разработка и реализация (в виде сервера и, при необходимости, клиента) некоторой сетевой службы (варианты, не вошедшие в предыдущие пункты).

Функциональность службы:

- установка на заданный порт (сервер)
- обнаружение сервера (клиент)
- ввод или генерация данных запроса (клиент)
- обработка запроса и возврат результата (сервер)
- прием и отображение ответа (клиент)
- контроль соединения, обработка ошибок и т.п.

Примеры служб:

- кодирование/декодирование данных (система счисления, алфавит/кодировка, транслитерация, азбука Морзе, простейшая криптография)
- обработка числовых данных
- сортировка, и т.п.

### **4 Сервер/клиент стандартной сетевой службы**

Реализация (возможно, упрощенная) функциональности некоторой существующей сетевой службы, протокола взаимодействия с ней.

Предполагается, что сервер службы (или клиент, в зависимости от того, что именно реализуется) используется готовый, уже функционирующий и доступный.

Примеры служб:

- echo (RFC-347) – сервер (клиент совсем уж неинтересно)
- time (RFC-686), daytime (RFC-867) – если реализуется клиент, то с хорошей интерпретацией полученных ответов
- systat (RFC-866)
- ping – использует протокол ICMP, а не IP/TCP/UDP
- почта (SMTP, POP, IMAP) – клиент
- FTP, TFTP
- HTTP, и т.п.

Во всех случаях ожидается допускается упрощение до минимально необходимого для поддержания работоспособности количества функций.

В качестве универсального готового TCP-клиента может выступать telnet.

### **5 Аналог сетевой утилиты**

Реализация утилиты, использующей сетевые протоколы или решающей задачи, связанные с сетью (управление, диагностика и проч.).

Функциональность – достаточна минимальная для демонстрации, интерфейс произвольный.

Примеры: nstat (порты локальной системы), сканер портов, сниффер (прослушивание передаваемых сообщений) и т.д.