Основы системного программирования

Дополнительная лабораторная работа
Процессы, потоки и средства межпроцессного взаимодействия

Написать на языке C две программы для ОС Linux:

1. сервер, поддерживающий заданный вариантом тип многозадачности (Табл. 2), транспортный протокол (Табл. 3) и прикладной протокол (Табл. 4);
2. клиент, поддерживающий заданный вариантом протокол и предназначенный для тестирования сервера.

Программы должны представлять собой консольные утилиты, настройка работы которых осуществляется путем передачи аргументов в строке запуска и/или с помощью переменных окружения:

```
lab2abcNXXXX_server [ опции ]
lab2abcNXXXX_client [ опции ]
```

Тип многозадачности серверной программы определяется согласно варианту из Табл. 2 путем подсчета количества букв в фамилии студента, выполняющего лабораторную работу. Транспортный протокол, поддерживаемый серверной и клиентской программами, определяется согласно варианту из Табл. 3 путем подсчета количества букв в имени студента, выполняющего лабораторную работу. Прикладной протокол, поддерживаемый серверной и клиентской программами, определяется согласно номеру варианта из Табл. 4. Серверная и клиентская программы должны поддерживать опции командной строки и переменные среды, перечисленные в Табл. 1 (расширение списка опций не запрещается).

Серверная программа должна обрабатывать поступление сигналов SIGINT, SIGTERM и SIGQUIT и корректно завершать работу при их поступлении. Также должен обрабатываться сигнал SIGUSR1, при поступлении которого серверная программа должна выводить в лог и в стандартный поток ошибок накопленную статистику — время работы, количество успешно обслуженных запросов и количество запросов, выполнение которых привело к ошибкам.

Серверная программа должна поддерживать запись отладочных сообщений и сообщений об ошибках в лог-файл, расположение которого определяется с помощью опции командной строки -l, соответствующей переменной окружения или умалчиваемым значением (см. Табл. 1). Сообщения, добавляемые в лог-файл, должны содержать временные метки в формате «ДД.ММ.ГГ чч:мм:сс». В лог также должны выводиться сообщения о поступлении нового запроса и завершении обслуживания очередного запроса.

Серверная программа должна поддерживать возможность запуска в режиме демона. Чтобы запустить её в этом режиме необходимо указать опцию -d (см. Табл. 1).

Адрес (IPv4) и порт, на котором сервер должен ожидать поступления запросов, передаются через опции -a и -p, если же они не заданы при запуске, тогда проверяются значения переменных окружения. Если же и переменные окружения не заданы, программа должна использовать значения по умолчанию (см. Табл. 1).

В случаях, когда задана и переменная среды, и опция командной строки, более приоритетной считается опция командной строки.

Поступивший запрос серверная программа должна передавать для обработки в рабочую задачу (процесс или поток в зависимости от варианта, см. Табл. 2) **и не блокироваться на время обработки запросов**.

Клиентская программа, с помощью которой выполняется проверка взаимодействия с сервером, должна позволять задать IP-адрес и номер порта и ввести запрос к серверу (в произвольной, но приемлемой форме; наиболее предпочтителен вариант ввода запроса через параметры командной строки).

Названия программ должны начинаться на lab2, далее должен следовать уникальный для варианта суффикс. Уникальный суффикс составляется из первых букв имени, отчества (если есть) и фамилии студента, выполняющего лабораторную работу. Далее следует номер группы студента. Используются строчные латинские буквы и арабские (в традиционном понимании, т. е. 0..9) цифры. Например, если студента, выполняющего лабораторную, зовут Петр Сергеевич Иванов, его группа — N3245, то имена программ должны быть `lab2psiN3245_server` и `lab2psiN3245_client`.

Проект (исходные коды, заголовочные файлы, Makefile и прочие файлы, необходимые для сборки) должен содержаться в отдельном каталоге с именем, совпадающим с названием программы (lab2abcNXXXX) и собираться с помощью стандартной утилиты make. Makefile должен поддерживать как минимум цели all и clean. Если для сборки проекта требуется что-то большее, чем `make all`, или для запуска и проверки проекта требуются какие-либо нетривиальные или неочевидные действия, то инструкции по сборке и запуску проекта следует добавить в файл README.txt в формате plain text и разместить его в каталоге проекта.

Порядок выполнения и сдачи лабораторной работы:

1. Выполнить задание, подготовить все файлы проекта, скомпилировать программу и библиотеку с флагами `-Wall -Wextra -Werror` и устранить все предупреждения и ошибки.
3. Протестировать программы (самое главное — на корректную обработку одновременных запросов), убедиться, что ошибок нет, в противном случае вернуться к пункту 2.
4. Выполнить запуск программ под управлением valgrind, убедиться, что утечки памяти отсутствуют. Если утечки есть, то сначала устранить их и вернуться к пункту 2. Если утечек нет, то сохранить отчет в файл valgrind.txt и добавить его в каталог проекта.
5. Скомпилировать программы с флагом `-O2` и повторно протестировать программы. В случае обнаружения ошибок вернуться к пункту 2.
6. Удалить все исполняемые и промежуточные файлы из папки проекта (`make clean`). В архиве должны остаться только файлы *.c, *.h, Makefile, README.txt, valgrind.txt.
7. Заархивировать папку проекта (`tar -czvf lab2abcNXXXX.tar.gz lab2abcNXXXX/`).
8. Отправить полученный архив на почту преподавателя, который ведет лабораторные (Гирику А.В. на itmo.osp@gmail.com, Грозову В.А. на va_groz@mail.ru), письмом с темой «*ОСП ЛР2 Фамилия Имя Отчество NXXXX*»
9. Дождаться ответа по почте или на лабораторном занятии, устранить возможные замечания (повторить с пункта 1).
10. Получить подтверждение от преподавателя, что лабораторная работа выполнена успешно, после чего подготовить отчет в электронной форме (состав отчета см. ниже).
11. Отправить архив с окончательным вариантом проекта и отчетом в формате pdf (не забыть про подпись на первой странице!) на почту преподавателя письмом с темой «*ОСП Отчет по ЛР2 Фамилия Имя Отчество NXXXX*». Файл отчета должен иметь название NXXX_ФамилияИО_ЛР2.pdf.
12. Получить некоторое количество вопросов по отчету от преподавателя и дать на них ответы (а может и не получить, если лабораторная выполнена на хорошем уровне и сомнений в знаниях студента у преподавателя не возникает). Получить от преподавателя подтверждение, что отчет принят.
13. Победа!

Отчет должен быть подготовлен в формате pdf и содержать:

- правильно оформленную титульную страницу (с подписью студента);
- задание;
- Make-файл;
- отчет valgrind со строчки «HEAP SUMMARY:»
- примеры работы программ (скриншоты);
- исходные тексты программ с комментариями.

Замечание 1. При выполнении лабораторной работы следует использовать функции стандартной библиотеки C и системные вызовы операционной системы. Использовать ввод-вывод в стиле C++ (классы ifstream/ofstream/…) запрещено. Использовать контейнеры и алгоритмы STL (<string>, <vector>, <map>, ...) запрещено. Использовать сторонние библиотеки (кроме pthread) запрещено.

Замечание 2. В программах должна присутствовать обработка ошибок: в случаях, если пользователь задал неверную комбинацию опций, указал файлы, которые невозможно открыть, и т.д. программа должна выдавать диагностическое сообщение на консоль (в стандартный поток ошибок и/или лог-файл), прежде чем завершиться. Получение неверно сформированного запроса не должно приводить к завершению программы-сервера.

Замечание 3. Категорически запрещается использовать статические массивы (с размерами, заданными на этапе компиляции) для входных данных. Для хранения входных данных необходимо использовать динамическую память и определять объем необходимой памяти в зависимости от объема входных данных.

Замечание 4. В вариантах, использующих в качестве транспортного протокола TCP, клиент должен закрывать соединение в сторону сервера после выдачи запроса, а сервер — после выполнения запроса клиента.

Замечание 5. Информационные сообщения и сообщения об ошибках выводятся серверной программой в лог-файл. Клиентская программа выводит информационные сообщения в стандартный поток вывода, сообщения об ошибках — в стандартный поток ошибок. С помощью определения переменной окружения `LAB2DEBUG` можно включить вывод отладочных сообщений программами в лог или в стандартный поток ошибок.

Замечание 6. Серверная и клиентская программы должны быть написаны таким образом, чтобы допускать одновременный запуск множества экземпляров (на разных интерфейсах и/или портах).

Замечание 7. В вариантах, предусматривающих запись чисел в десятичной или шестнадцатеричной системах счисления, признаком использования шестнадцатеричной системы счисления служит префикс «0x».

Замечание 8. Пробельными символами в лабораторной работе считаются собственно пробел SP (\x20), горизонтальная табуляция HT (\x09 или \t), вертикальная табуляция VT (\x0B), прогон страницы FF (\x0C) и возврат каретки CR (\x0D или \r). Символ перевода строки LF (\ x0A или \n) также относится к пробельным, но в большинстве вариантов имеет специальный смысл (маркер конца строки) и поэтому не может использоваться для разделения слов в строке.

Замечание 9. Подсчет букв в имени и фамилии выполняется в русской транскрипции.

Замечание 10. Для обработки сигналов не следует использовать функцию signal(). Вся обработка сигналов должна выполняться только с помощью функций интерфейса POSIX sig*set()/sigprocmask()/sigaction() и т. д.

Замечание 11. Несмотря на то, что для компиляции программ необходимо использовать компилятор gcc, использования расширений GNU C желательно по возможности избегать и ориентироваться на использование стандарта C11 или более позднего.

Таблица 1. Переменные среды и опции командной строки, поддерживаемые программой-сервером и программой-клиентом

Опция	Переменная среды	Назначение	Значение по умолчанию	Поддержи-вается
-w N	LAB2WAIT	Имитация работы путем приостановки	0	Сервером

		обслуживающего запрос процесса/потока на N секунд. Если опция не задана, обслуживать запрос без задержки.		
-d		Работа в режиме демона.		Сервером
-l /path/to/log	LAB2LOGFILE	Путь к лог-файлу.	/tmp/lab2.log	Сервером
-a ip	LAB2ADDR	Адрес, на котором слушает сервер и к которому подключается клиент.		Сервером и клиентом
-p port	LAB2PORT	Порт, на котором слушает сервер и к которому подключается клиент.		Сервером и клиентом
-v		Вывод версии программы.		Сервером и клиентом
-h		Вывод справки по опциям.		Сервером и клиентом
	LAB2DEBUG	Включение вывода отладочной информации.		Сервером и клиентом

Таблица 2. Тип многозадачности программы-сервера

Вариант (условие)	Тип многозадачности
Количество букв в **фамилии** студента, выполняющего работу, **нечетное**.	Многопроцессность (создание рабочих задач с помощью вызова fork).
Количество букв в в **фамилии** студента, выполняющего работу, **четное**.	Многопоточность (создание рабочих потоков средствами библиотеки libpthread).

Таблица 3. Транспортный протокол, поддерживаемый программой-сервером и программой-клиентом

Вариант (условие)	Транспортный протокол
Количество букв в **имени** студента, выполняющего работу, **нечетное**.	UDP
Количество букв в в **имени** студента, выполняющего работу, **четное**.	TCP

Таблица 4. Прикладной протокол, поддерживаемый программой-сервером и программой-клиентом

Вариант (номер)	Описание прикладного протокола
1	*Запрос*: строка, содержащая список целых чисел в десятичной или шестнадцатеричной системах счисления, разделенных пробельными символами (возможно, несколькими подряд), завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая те числа из списка, которые являются простыми (с сохранением порядка следования в запросе, разделитель — пробел), завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
2	*Запрос*: 2 строки, каждая из которых завершается символом LF. Первая строка содержит список вещественных чисел в обычном или научном формате, разделенных пробелами или символами табуляции (возможно, несколькими подряд). Вторая строка содержит способ округления (в сторону минус бесконечности, в сторону плюс бесконечности, в сторону нуля). *Ответ, если ошибок не было*: строка, содержащая округленные по заданному способу вещественные числа (с сохранением порядка следования в запросе, разделитель — пробел), завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где

	N — код ошибки.
3	*Запрос*: 7 байтов, первые 6 байтов содержат некоторый MAC-адрес, последний байт кодирует тип запроса (добавить, удалить, проверить). *Ответ, если ошибок не было*: для операций добавления и удаления — 1 байт со значением 0; для операции проверки — 1 байт со значением 0, если указанный MAC-адрес не добавлен, и со значением 1 — если добавлен. *Ответ, если были ошибки*: 1 байт со значением, отличным от 0 и 1, кодирующим ошибку. *Дополнительные требования*: добавленные MAC-адреса должны сохраняться сервером между перезапусками.
4	*Запрос*: 24 байта, содержащих 3 пары вещественных чисел одинарной точности (float), задающих координаты вершин некоторого треугольника на плоскости. *Ответ, если ошибок не было*: 5 байтов, первый из которых — 0, а остальные четыре содержат вещественное число одинарной точности, представляющее собой периметр треугольника, заданного координатами в запросе. *Ответ, если были ошибки*: 1 байт со значением, отличным от 0, кодирующим ошибку.
5	*Запрос*: строка, содержащая список целых чисел в десятичной или шестнадцатеричной системах счисления, разделенных пробелами или символами табуляции (возможно, несколькими подряд), завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая отсортированный список чисел (разделитель — пробел, критерий сортировки — по убыванию количества разрядов в десятичном представлении числа), завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
6	*Запрос*: 2 строки, каждая из которых завершается символом LF. Первая строка содержит команду (добавить, удалить, получить). Вторая строка содержит список вещественных чисел в обычном или научном формате, разделенных пробелами или символами табуляции (возможно, несколькими подряд) в случае команд добавления или удаления, и целое число N — в случае команды получения. *Ответ, если ошибок не было*: в случае команд добавления и удаления — строка «OK», заверщающаяся символом LF; в случае команды получения — строка, содержащая N случайно выбранных из множества добавленных на момент поступления запроса чисел, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки. *Дополнительные требования*: множество добавленных чисел должно сохраняться сервером между перезапусками.
7	*Запрос*: строка, содержащая целое число N в десятичной записи и два вещественных числа F и T в обычном или научном формате, разделенных пробельными символами, завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая N случайных, не повторяющихся с момента первого запуска вещественных чисел из диапазона F .. T, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки. *Дополнительные требования*: история сгенерированных чисел должна сохраняться между перезапусками.
8	*Запрос*: строка, содержащая слова, разделенные пробельными символами, завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая анаграммы слов, переданных в запросе, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
9	*Запрос*: строка, содержащая целые числа в десятичной или шестнадцатеричной записи, разделенные пробельными символами, завершающаяся символом LF. *Ответ, если ошибок не было*: строка «ALL», если все числа являются какими-либо членами последовательности Фибоначчи (не обязательно следующими

	подряд); строка «SOME», если некоторые из чисел являются членами последовательности Фибоначчи; «NONE», если членов последовательности Фибоначчи среди чисел, переданных в запросе, не было. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
10	*Запрос*: строка, завершающаяся символом LF, содержащая либо «GET имя_переменной», либо «SET имя_переменной=значение», либо «DELETE имя_переменной», причем в имени переменной не может содержаться знака равенства. *Ответ, если ошибок не было*: на запрос «GET имя_переменной» возвращается строка вида «FOUND имя_переменной=значение» или строка вида «NOT FOUND»; на запрос «SET имя_переменной=значение» возвращается строка «OK»; на запрос «DELETE имя_переменной» возвращается строка «OK»; в каждом случае строки завершаются символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки. *Дополнительные требования*: набор переменных и их значения должны сохраняться сервером между перезапусками.
11	*Запрос*: 5 байтов, первые 4 байта содержат некоторый IP-адрес (IPv4), последний байт кодирует тип запроса (добавить, удалить, проверить). *Ответ, если ошибок не было*: для операций добавления и удаления — 1 байт со значением 0; для операции проверки — 1 байт со значением 0, если указанный IP-адрес не добавлен, и со значением 1 — если добавлен. *Ответ, если были ошибки*: 1 байт со значением, отличным от 0 и 1, кодирующим ошибку. *Дополнительные требования*: добавленные IP-адреса должны сохраняться сервером между перезапусками.
12	*Запрос*: 2 строки, заврешающихся символом LF. Первая содержит двухбуквенный селектор языка («EN» либо «RU»), вторая — целое число в десятичной системе счисления. *Ответ, если ошибок не было*: строка, содержащая число, переданное в запросе, прописью (обеспечить поддержку как минимум до дециллиона по короткой шкале включительно), завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
13	*Запрос*: строка, содержащая целые числа в десятичной или шестнадцатеричной записи, разделенные пробельными символами, завершающаяся символом LF. *Ответ, если ошибок не было*: строка «ALL», если все числа являются центральными многоугольными числами; строка «SOME», если некоторые из чисел являются центральными многоугольными числами; «NONE», если среди чисел, переданных в запросе, центральных многоугольных чисел не было. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
14	*Запрос*: строка, содержащая список из трех пар вещественных чисел в обычном или научном формате, разделенных пробелами или символами табуляции (возможно, несколькими подряд), завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая вещественное число, представляющее собой площадь треугольника, заданного координатами на плоскости (парами чисел в запросе), завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
15	*Запрос*: строка, содержащая список целых чисел в десятичной или шестнадцатеричной системах счисления, разделенных пробелами или символами табуляции (возможно, несколькими подряд), завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая отсортированный список чисел (разделитель — пробел, критерий сортировки — по возрастанию количества единичных разрядов в двоичном представлении числа), завершающаяся символом LF.

	Ответ, если были ошибки: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
16	*Запрос*: от 2 до 256 байтов, первый из которых содержит количество байтов данных. *Ответ, если ошибок не было*: строка, содержащая двоичное представление каждого байта данных (байты отделяются друг от друга пробелами), завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
17	*Запрос*: строка, содержащая арифметическое выражение с целыми и/или вещественными числами и операциями сложения, вычитания, деления, умножения и скобками, завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая результат вычисления выражения, переданного в запросе, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки. *Указание*: для вычисления арифметических выражений можно воспользоваться стандартными утилитами dc или bc.
18	*Запрос*: от 3 до 65536 байтов, первые два из которых кодируют количество байтов данных, остальные представляют собой байты данных. *Ответ, если ошибок не было*: 3 байта, первый из которых нулевой, а другие два содержат контрольную сумму байтов данных, переданных в запросе, подсчитанную по алгоритму CRC-16-ANSI. *Ответ, если были ошибки*: 3 байта, первый из котоырх ненулевой и содержит код ошибки, два других — нулевые.
19	*Запрос*: строка, содержащая имя хоста или домена, завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая список IP-адресов, в которые разрешается переданное в запросе имя, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
20	*Запрос*: строка, содержащая 16 цифр номера банковской карты (допускаются дефисы или пробельные символы между группами четырех цифр), завершающаяся символом LF. *Ответ, если ошибок не было*: «OK», если номер корректный согласно алгоритму проверки ISIN, «FAILED», если номер некорректный. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
21	*Запрос*: строка «PUSH число», где число — вещественное или целое число, либо строка «POP». Строки завершаются символом LF. *Ответ, если ошибок не было*: строка «OK» в случае запроса «PUSH», помещающего число в стек, или строка, содержащая извлеченное из стека число в случае запроса «POP». Строки завершаются символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки. *Дополнительные требования*: по умолчанию для каждого клиента сервер должен поддерживать свой стек (клиенты идентифицируются по IP-адресу отправителя запроса). Необходимо реализовать возможность запустить сервер в режиме общего для всех клиентов стека.
22	*Запрос*: строка, содержащая 11 цифр номера СНИЛС (допускаются дефисы и пробельные символы между группами цифр), завершающаяся символом LF. *Ответ, если ошибок не было*: «OK», если номер корректный согласно алгоритму проверки контрольного числа СНИЛС, «FAILED», если номер некорректный. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
23	*Запрос*: строка, содержащая арифметическое выражение в инфиксной записи, включающее целые и вещественные числа, знаки арифметических операций и скобки, завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая выражение из запроса,

	преобразованное в постфиксную (обратную польскую) запись, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
24	*Запрос*: строка, содержащая выражение вида *OP X1 ... XN*, завершающаяся символом LF, где *OP* — одна из поразрядных опер аций AND, OR, XOR, NAND или NOR, *X1 ... XN* — целочисленные операнды, записанные в десятичной или шестнадцатеричной форме, разделенные пробельными символами. *Ответ, если ошибок не было*: строка, содержащая результат операции из запроса, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
25	*Запрос*: строка, содержащая выражение вида «RGB R G B» или «HSL H S L», завершающаяся символом LF, где R, G, B, H, S, L — целые числа, компоненты соответствующих цветовых пространств, записанные в десятичной или шестнадцатеричной форме, разделенные пробельными символами. *Ответ, если ошибок не было*: строка, содержащая результат преобразования из одного цветового пространства в другое в таком же формате, что и строка запроса, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
26	*Запрос*: строка, содержащая разделенные пробельными символами числа, записанные арабскими цифрами (в десятичной системе счисления) и римскими цифрами, завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая разделенные пробелами переводы чисел (из десятичной системы счисления в римскую и наоборот), завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
27	*Запрос*: от 3 до 65536 байтов, первые два из которых кодируют количество байтов данных, остальные представляют собой байты данных. *Ответ, если ошибок не было*: строка, содержащая представление байтов данных, переданных в запросе, в кодировке base64, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
28	*Запрос*: строка вида «CIDR1 in CIDR2», завершающаяся символом LF, где CIDR1 и CIDR2 — IPv4-сети, записанные в формате CIDR. *Ответ, если ошибок не было*: завершающаяся символом LF строка, содержащая «YES», если CIDR1 является подсетью CIDR2, или «NO», если не является. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
29	*Запрос*: строка, содержащая 17 знаков номера VIN, завершающаяся символом LF. *Ответ, если ошибок не было*: «OK», если номер корректный согласно алгоритму проверки, утвержденному стандартом ISO 3779-1983, «FAILED», если номер некорректный. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
30	*Запрос*: строка «PUT число», где число — вещественное или целое число, либо строка «GET». Строки завершаются символом LF. *Ответ, если ошибок не было*: строка «OK» в случае запроса «PUT», помещающего число в очередь, или строка, содержащая извлеченное из очереди число в случае запроса «GET». Строки завершаются символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки. *Дополнительные требования*: по умолчанию сервер поддерживает общую для всех клиентов очередь. Необходимо реализовать возможность запустить сервер в режиме отдельной очереди для каждого клиента (клиенты идентифицируются по IP-адресу отправителя запроса).

31	*Запрос*: строка, содержащая список пар вещественных чисел в обычном или научном формате, разделенных пробелами или символами табуляции (возможно, несколькими подряд), завершающаяся символом LF. *Ответ, если ошибок не было*: строка, содержащая вещественное число, представляющее собой площадь под ломаной линией, заданной координатами на плоскости — отсортированными по величине абсциссы парами чисел в запросе, завершающаяся символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки.
32	*Запрос*: строка «ADD число», где число — вещественное или целое число, либо строка «GET». Строки завершаются символом LF. *Ответ, если ошибок не было*: строка «OK» в случае запроса «ADD», помещающего число в множество, или строка, содержащая случайное извлеченное из множества число в случае запроса «GET». Строки завершаются символом LF. *Ответ, если были ошибки*: строка «ERROR N», завершающаяся символом LF, где N — код ошибки. *Дополнительные требования*: по умолчанию сервер поддерживает общее для всех клиентов множество. Необходимо реализовать возможность запустить сервер в режиме отдельного множества для каждого клиента (клиенты идентифицируются по IP-адресу отправителя запроса).