

Федеральное государственное автономное образовательное учреждение
высшего образования
"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
"ВЫСШАЯ ШКОЛА ЭКОНОМИКИ"

Факультет компьютерных наук
Департамент программной инженерии

ИНДИВИДУАЛЬНОЙ ЗАДАНИЕ №3 ПО ДИСЦИПЛИНЕ
«ОПЕРАЦИОННЫЕ СИСТЕМЫ»

Сетевые взаимодействия с применением
транспортного протокола TCP

Москва 2023

СПИСОК СОКРАЩЕНИЙ

- API — Application Programming Interface. Интерфейс Программирования приложения — программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими.
- bash — Bourne Again shell. Одна из наиболее популярных командной оболочки UNIX. Популярна в Linux. Часто используется в качестве предустановленной командной оболочки.
- CLI — Command Line Interface. Интерфейс командной строки
- IDE — Integrated Development Environment. Интегрированная среда разработки — система программных средств, используемая программистами для разработки программного обеспечения.
- ISA — Instruction Set Architecture
- POSIX — Portable Operating System Interface. Переносимый интерфейс операционных систем — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов.
- ОС — операционная система
- ЯП — язык программирования
- ЯПВУ — язык программирования высокого уровня

Оглавление

Индивидуальные задания	4
Общее условие	4
Варианты заданий	4
Требование к выполнению и оценке	34

Индивидуальные задания

Общее условие

Цели и задачи: Изучить работу с транспортным протоколом TCP. Научиться разбивать задачу на части, для последующего их выполнения серверами и клиентами.

Архитектура «клиент–сервер» широко используется при решении разнообразных прикладных задач. Существуют различные подходы к организации таких приложений с использованием для организации серверов и клиентов как процессов, так и потоков.

В ходе выполнения задания необходимо осуществить разделить функции, выполняемые отдельными компонентами и организовать их взаимодействие, необходимое для выполнения заданной прикладной задачи.

Варианты заданий

1. **Задача о парикмахере.** В тихом городке есть парикмахерская. Салон парикмахерской мал, работать в нем может только один парикмахер, обслуживающий одного посетителя. Есть несколько стульев для ожидания в очереди. Парикмахер всю жизнь обслуживает посетителей. Когда в салоне никого нет, он спит в кресле. Когда посетитель приходит и видит спящего парикмахера, он будет его, садится в кресло, и сидит в нем, пока парикмахер обслуживает его. Если посетитель приходит, а парикмахер занят, то он встает в очередь, садится на свободный стул и «засыпает». После стрижки парикмахер сам провожает посетителя. Если есть ожидающие посетители, то парикмахер будит одного из них, ждет пока тот сядет в кресло парикмахера и начинает стрижку. Если никого нет, он снова садится в свое кресло и засыпает до прихода посетителя.

Количество стульев для ожидания в очереди ограничено числом N . Если стульев не хватает, то пришедший посетитель уходит.

Создать клиент–серверное приложение, моделирующее рабочий день парикмахерской.

Парикмахера и парикмахерскую моделировать в виде сервера. Каждого из посетителей моделировать в виде отдельного клиента. Клиенты могут запускаться независимо друг от друга как вручную, так и с использованием скриптов, или других программ, запускающих их в фоновом режиме. В этом случае предусмотреть задержку, задающую паузу между запуском клиентов.



2. **Задача о Винни-Пухе и правильных пчелах.** В одном лесу живут N пчел и один медведь, которые используют один горшок меда, вместимостью H глотков. Сначала горшок пустой. Пока горшок не наполнится, медведь спит. Как только горшок заполняется, медведь просыпается и съедает весь мед, после чего снова засыпает. Каждая пчела многократно собирает по одному глотку меда и кладет его в горшок. Пчела, которая приносит последнюю порцию меда, будит медведя.

Создать клиент–серверное приложение, моделирующее поведение пчел и медведя.

Медведя и улей с пчелами представить в виде отдельных клиентов. Сервер обрабатывает поступающие от них сообщения и передает их клиентам в соответствии с установленными выше правилами.



3. Задача о Винни-Пухе – 2 или неправильные пчелы. N пчел живет в улье. Каждая пчела может собирать мед или сторожить улей ($N > 3$). Пчела не покинет улей, если кроме нее в нем нет других пчел. Каждая пчела приносит за раз одну порцию меда. Всего в улей может войти тридцать порций меда. Винни-Пух спит пока меда в улье меньше половины, но как только его становится достаточно, он просыпается и пытается достать весь мед из улья. Если в улье находится меньше трех пчел, Винни-Пух забирает мед, убегает, съедает мед и снова засыпает. Если пчел три или больше, то они кусают Винни-Пуха, он убегает, лечит укус некоторое время, а затем снова бежит за медом.

Создать клиент–серверное приложение, моделирующее поведение пчел и медведя.

Медведя и улей с пчелами представить в виде отдельных клиентов. Сервер обрабатывает поступающие от них сообщения и передает их клиентам в соответствии с установленными выше правилами.

4. Задача о Винни-Пухе – 3 или мстительные пчелы. Неправильные пчелы, подсчитав в конце месяца убытки от наличия в лесу Винни-Пуха, решили разыскать его и наказать в назидание всем другим любителям сладкого. Для поисков медведя они поделили лес на участки, каждый из которых прочесывает одна стая неправильных пчел. В случае нахождения медведя на своем участке стая проводит показательное наказание и возвращается в улей. Если участок прочесан, а Винни-Пух на нем не обнаружен, стая



также возвращается в улей. Там она получает информацию об еще участках, которые еще не исследованы и снова улетает. Или прекращает поиски, узнав, что Винни-Пух наказан.

Создать клиент–серверное приложение, моделирующее поведение пчел.

Каждая стая — отдельный клиент, взаимодействующий с сервером. Сервер обрабатывает поступающие сообщения и передает их клиентам в соответствии с установленными выше правилами.



5. Задача о читателях и писателях. Базу данных, представленную массивом целых положительных чисел, разделяют два типа процессов: **N** читателей и **K** писателей. Читатели периодически просматривают случайные записи базы данных и выводят номер своей записи (например, PID), индекс записи, ее значение, а также

вычисленное значение числа Фибоначчи. Писатели изменяют случайные записи на случайное число и также выводят информацию о своем номере, индексе записи, старом значении и новом значении. Предполагается, что в начале БД находится в непротиворечивом состоянии (все числа отсортированы, например, по возрастанию). Каждая отдельная новая запись переводит БД из одного непротиворечивого состояния в другое (то есть, новая сортировка может поменять индексы записей или переставить числа). Для предотвращения взаимного влияния транзакций процесс–писатель должен иметь исключительный доступ к БД. Если к БД не обращается ни один из процессов–писателей, то выполнять транзакции могут одновременно сколько угодно читателей.

Создать клиент–серверное приложение с процессами–пи-сателями и процессами–читателями.

Сервер моделирует базу данных. Все писатели и все читатели – два разных клиента, **в каждом из которых** возможна конкуренция параллельных процессов или потоков. Каждый процесс – это отдельный писатель или отдельный читатель *внутри сервера*.

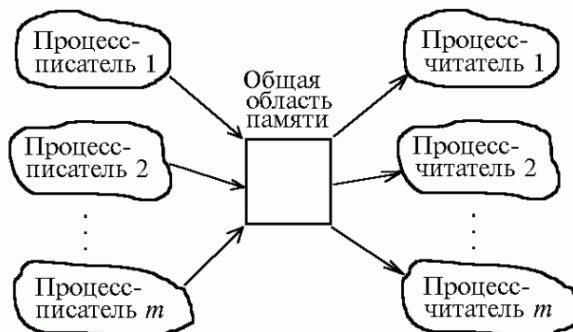


6. **Задача о читателях и писателях («грязное чтение»).** Базу данных, представленную массивом целых положительных чисел, разделяют два типа процессов: **N** читателей и **K** писателей. Читатели периодически просматривают случайные записи базы данных и выводя номер свой номер (например, PID), индекс записи, ее значение, а также вычисленное значение факториала. Писатели

изменяют случайные записи на случайное число и также выводят информацию о своем номере, индексе записи, старом значении и новом значении. Предполагается, что в начале БД находится в непротиворечивом состоянии (все числа отсортированы). Каждая отдельная новая запись переводит БД из одного непротиворечивого состояния в другое (то есть, новая сортировка может поменять индексы записей или переставить числа). Транзакции выполняются в режиме «грязного чтения». То есть, процесс-писатель не может получить доступ к БД только в том случае, если ее уже занял другой процесс-писатель, а процессы-читатели ему не мешают обратиться к БД. Поэтому он может изменять базу данных, когда в ней находятся читатели.

Создать клиент–серверное приложение с процессами–пи-сателями и процессами–читателями.

Сервер моделирует базу данных. Все писатели и все читатели – два разных клиента, **в каждом из которых** возможна конкуренция параллельных процессов или потоков. Каждый процесс – это отдельный писатель или отдельный читатель *внутри сервера*.



7. Задача о читателях и писателях («подтвержденное чтение»). Базу данных, представленную массивом целых положительных чисел, разделяют два типа процессов: **N** читателей и **K** писателей. Читатели периодически просматривают случайные записи базы данных и выводя номер свой номер (например, PID), индекс записи, ее значение, а также вычисленное значение, которое является произведением числа на номер записи. Писатели изменяют случайные записи

на случайное число и также выводят информацию о своем номере, индексе записи, старом значении и новом значении. Предполагается, что в начале БД находится в непротиворечивом состоянии (все числа отсортированы). Каждая отдельная новая запись переводит БД из одного непротиворечивого состояния в другое (то есть, новая сортировка может поменять индексы записей или переставить числа). Транзакции выполняются в режиме «подтвержденного чтения», то есть процесс-писатель не может получить доступ к БД в том случае, если ее занял другой процесс-писатель или процесс-читатель. К БД может обратиться одновременно сколько угодно процессов-читателей. Процесс читатель получает доступ к БД, даже если ее уже занял процесс-писатель.

*Создать клиент–серверное приложение с процессами–пи-
сателями и процессами–читателями.*

*Сервер моделирует базу данных. Все писатели и все читатели –
два разных клиента, в каждом из которых возможна конку-
ренция параллельных процессов или потоков. Каждый процесс –
это отдельный писатель или отдельный читатель внутри сер-
вера.*

8. **Задача об обедающих философах.** Это классическая задача на взаимодействие параллельных процессов. **Пять** философов сидят возле круглого стола. Они проводят жизнь, чередуя приемы пищи и размышления. В центре стола находится большое блюдо спагетти. Спагетти длинные и запутанные, философам тяжело управляться с ними, поэтому каждый из них, что бы поесть, должен пользоваться двумя вилками. К несчастью, философам дали только **пять** вилок. Между каждой парой философов лежит одна вилка. Поэтому эти высококультурные и предельно вежливые люди договорились, что каждый будет пользоваться только теми вилками, которые лежат рядом с ним (слева и справа).

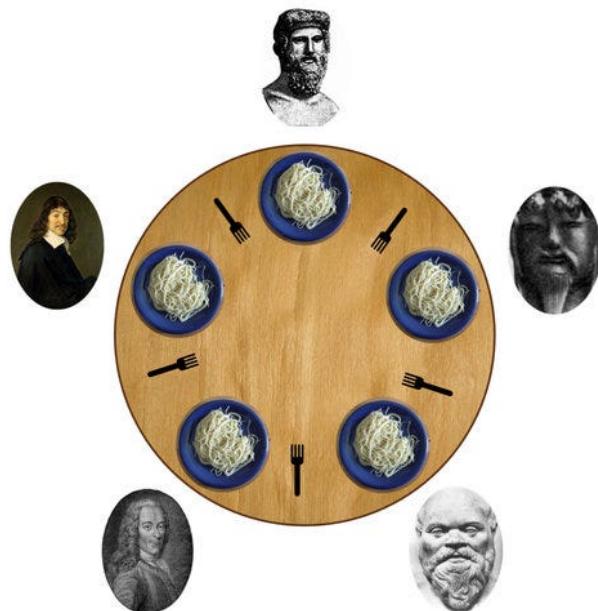
*Написать клиент–серверную программу, моделирующую
поведение философов с помощью семафоров.*

Философы являются отдельными клиентами, синхронизируемы-

ми посредством сервера, который фиксирует состояние стола и организует взаимодействие с философами.

Программа должна избегать фатальной ситуации, в которой все философы голодны, но ни один из них не может взять обе вилки (например, каждый из философов держит по одной вилке и не хочет отдавать ее). Время, которое отводится на прием пищи и размышление задается случайно в некотором разумном для наблюдения из вне диапазоне.

Решение должно быть симметричным, то есть все процессы – философы должны выполнять один и тот же код (являясь равноправными процессами).



9. Задача о каннибалах. Племя из N дикарей ест вместе из большого горшка, который вмещает M кусков тушеного миссионера. Когда дикарь хочет обедать, он ест из горшка один кусок, если только горшок не пуст, иначе дикарь будит повара и ждет, пока тот не наполнит горшок. Повар, сварив обед, засыпает.

Создать клиент-серверное приложение, моделирующее обед дикарей.

Каждый дикарь должен быть представлен отдельным клиентом. Повар – сервер. Максимальное количество дикарей определяется

при запуске сервера. То есть, сервер с ограничением на число клиентов.

Повар и каждый из дикарей задаются отдельными процессами.



10. **Задача о курильщиках.** Есть три процесса—курильщика и один процесс—посредник. Курильщик непрерывно скручивает сигареты и курит их. Чтобы скрутить сигарету, нужны табак, бумага и спички. У одного курильщика есть табак, у второго — бумага, а у третьего — спички. Посредник через **некоторое случайное время** кладет на стол по два разных случайных компонента. Тот Курильщик, у которого есть третий компонент, забирает компоненты со стола, скручивает сигарету и курит **некоторое случайное время**. Посредник через отведенный ему интервал времени, который может быть больше или меньше времени курения, выкладывает следующий набор. Если курильщик, которому нужен этот набор, свободен, то он начинает курить. Если курильщик еще курит, то процесс передачи дожидается окончания курения. Затем процесс повторяется. В принципе возможна ситуация, когда все три курильщика могут курить одновременно или все могут ждать, когда посредник соизволит выложить очередной набор.

Создать клиент–серверное приложение, моделирующее поведение курильщиков и посредника.

Каждый курильщик — клиент. Посредник — сервер.

11. **Военная задача.** Анчуария и Тарантерия — два крохотных латиноамериканских государства, затерянных в южных Андах. Диктатор Анчуарии, дон Федерико, объявил войну диктатору Таран-



терии, дону Эрнандо. У обоих диктаторов очень мало солдат, но очень много снарядов для минометов, привезенных с последней американской гуманитарной помощью. Поэтому армии обеих сторон просто обстреливают наугад территорию противника, надеясь поразить что-нибудь ценное. Стрельба ведется по очереди до тех пор, пока либо не будут уничтожены все **K** цели (каждая стоимостью C_i), либо стоимость **N** потраченных снарядов не превысит суммарную стоимость всего того, что ими можно уничтожить. Размер каждого государства $X \times Y$ клеток. Стрельба ведется асинхронно и параллельно (а не последовательно) через случайное время, задаваемое для обеих государств посредством одинаковых алгоритмов генерации случайных чисел. То есть, числа изменяются в одном и том же диапазоне, но на каждом шаге стрельбы они могут отличаться для разных государств.

Создать клиент–серверное приложение, моделирующее военные действия.

Каждая страна – отдельный клиент. Сервер отвечает за прием координат от клиентов и передает эти координаты другим клиентам. Он также получает информацию от клиентов об их уничтожении. Расположение минометов порождается каждым клиентом случайно.

Обратить особое внимание на отображение состояния государств в процессе моделирования.



- 12. Задача о супермаркете.** В супермаркете работают **два** кассира, покупатели заходят в супермаркет, делают покупки и становятся в очередь к случайному кассиру. Пока очередь пустая, кассир «спит». Как только появляется покупатель, кассир «просыпается». Покупатель ожидает в очереди, пока не подойдет к кассиру. Очереди имеют ограниченную длину **N**. Если она длиннее, то покупатель не встает ни в одну из очередей и уходит. Если одна из очередей заполнена, то покупатель встает в другую.

Создать клиент–серверное приложение, моделирующее рабочий день супермаркета.

Каждого кассира реализовать в виде отдельного клиента. Отдельный клиент задает всех покупателей. Сервер распределяет покупателей по очередям к кассирам, передавая сообщения соответствующим клиентам.

- 13. Первая задача о магазине.** В магазине работают **три** отдела, каждый отдел обслуживает **один** продавец. Покупатель, зайдя в магазин, делает покупки в одном или нескольких произвольных отделах, обходя их в **произвольном (случайном)** порядке. Если в выбранном отделе продавец не свободен, покупатель становится в очередь и ожидает, пока продавец не освободится.

Создать многопроцессное приложение, моделирующее рабочий день магазина.

Каждого покупателя и продавцов моделировать отдельными процессами.



Создать клиент–серверное приложение, моделирующее рабочий день магазина.

Каждый продавец – клиент, обслуживающий покупателя случайное время, Сервер создает покупателей, «сопровождает» их случайнym образом по отделам и обеспечивает завершение обхода магазина.

Размер очереди не оговаривается. Считается, что для данной задачи она не ограничена (но моделирование должно быть в разумных пределах).



14. Вторая Задача о магазине (забывчивые покупатели). В магазине работают **два** отдела, каждый отдел обладает уникальным

ассортиментом. В каждом отделе работает один продавец. В магазин ходят исключительно забывчивые покупатели, поэтому каждый покупатель носит с собой список из **K** товаров, которые желает купить. Покупатель приобретает товары точно в том порядке, в каком они записаны в его списке. Число товаров в списке от одного до пяти. При этом товары в списке расположены в **случайном** порядке, что заставляет покупателя многократно переходить от отдела к отделу, если это требуется для совершения покупок. Продавец может обслужить только одного покупателя за раз. Покупатель, вставший в очередь, засыпает пока не дойдет до продавца. Продавец засыпает, если в его отделе нет покупателей, и просыпается, если появится хотя бы один.

Создать клиент–серверное приложение, моделирующее работу магазина в течение рабочего дня.

Каждый продавец – клиент. Покупатели изначально порождаются отдельным клиентом. Сервер обеспечивает взаимодействие покупателей и клиентов.

15. **Задача о больнице.** В больнице **два** дежурных врача принимают пациентов, выслушивают их жалобы и отправляют или к стоматологу, или к хирургу, или к терапевту. Стоматолог, хирург и терапевт лечат пациентов. Каждый врач может принять только одного пациента за раз. Пациенты стоят в очереди к врачам и никогда их не покидают.

Создать клиент–серверное приложение, моделирующее рабочий день клиники.

Каждого врача и очередь к нему реализовать как независимого клиента. Поступающие пациенты формируются отдельным общим клиентом. Сервер перераспределяет пациентов по очередям и отслеживает общее взаимодействие клиентов.

16. **Задача о социалистической гостинице.** В гостинице **10** одноместных номеров. Клиенты гостиницы снимают номер на одни или несколько суток (задается при создании клиента). Если в гостинице нет свободных номеров, клиенты не уходят, а устраиваются на



рядом с гостиницей на скамейках и ждут в порядке очереди, пока любой из номеров не освободится (других гостиниц в городе нет).

Создать клиент–серверное приложение, моделирующее работу гостиницы.

Сервер — это гостиница. Прибывающие гости могут порождать отдельным общим клиентом. Другой клиент — это скамейки, образующие очередь ожидающих гостей.



17. Задача о капиталистической гостинице. В гостинице 5 номеров с ценой 2000 УЕ, 5 номеров с ценой 4000 УЕ и 5 номеров с ценой 6000 УЕ. Клиент, зашедший в гостиницу, обладает некоторой (случайной) суммой и получает номер по своим финансовым возможностям, если тот свободен. При наличии денег и разных по стоимости номеров он выбирает **случайный** номер. Если доступных по цене свободных номеров нет, клиент уходит искать ночлег в другое место. Клиенты порождаются динамически и уничтожаются при освобождении номера или уходе из гостиницы при невозможности оплаты.

Создать клиент–серверное приложение, моделирующее работу гостиницы.

Гостиница — сервер. Каждого гостя реализовать в виде отдельного клиента, порождаемого вручную. Можно запустить скрипт, порождающий сразу множество гостей в фоновом режиме со случайной задержкой между их приходом в гостиницу.

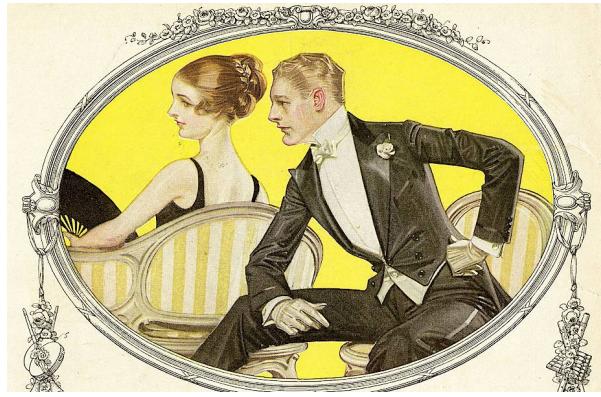


18. **Задача о гендерной гостинице (леди и джентльмены).** В гостинице 7 номеров рассчитаны на одного человека и 10 номеров рассчитаны на двух человек. В гостиницу случайно приходят клиенты: леди и джентльмены. И конечно они могут провести ночь в номере только с представителем своего пола. Если для клиента не находится подходящего номера, он уходит искать ночлег в другое место. Клиенты порождаются динамически и уничтожаются при освобождении номера или уходе из гостиницы при невозможности поселиться.

Создать клиент–серверное приложение, моделирующее работу гостиницы.

Гостиница — сервер. Каждого гостя реализовать в виде отдельного клиента, порождаемого вручную. Можно запустить скрипт, порождающий сразу множество гостей в фоновом режиме со случайной задержкой между их приходом в гостиницу.

19. **Задача об умной клумбе.** На клумбе растет **20** цветов, за ними непрерывно следят два садовника и поливают увядшие цветы, при



этом оба садовника очень боятся полить один и тот же цветок, который еще не начал вянуть. Иначе он пропадет из-за перелива.

Создать клиент–серверное приложение, моделирующее состояние цветов на клумбе и действия садовников.

Клумба – сервер, который следит за состоянием всех цветков и информирует, который из них полит, увядает (нужно полить), засох или перелит и сгнил (бесполезно поливать). Каждый садовник является отдельным клиентом.



20. **Задача об умных цветах.** На клумбе растет **10** цветов, за ними непрерывно следят два садовника и поливают увядшие цветы, при этом оба садовника очень боятся полить один и тот же цветок, который еще не начал вянуть. Иначе он пропадет из-за перелива.

Создать клиент–серверное приложение, моделирующее состояние цветов на клумбе и действия садовников.

Сервер используется для обмена информацией между садовниками и клумбой. Клумба – клиент, отслеживающий состояния

всех цветов. Каждый садовник — отдельный клиент.

21. **Задача о нелюдимых садовниках.** Имеется пустой участок земли (двумерный массив размером $M \times N$) и план сада, разбитого на отдельные квадраты. От 10 до 30 процентов (задается случайно) площади сада заняты прудами или камнями. То есть недоступны для ухаживания. Эти квадраты располагаются на плане произвольным (случайным) образом. Ухаживание за садом выполняют два садовника, которые не хотят встречаться друг другом (то есть, одновременно появляться в одном и том же квадрате). Первый садовник начинает работу с верхнего левого угла сада и перемещается слева направо, сделав ряд, он спускается вниз и идет в обратном направлении, пропуская обработанные участки. Второй садовник начинает работу с нижнего правого угла сада и перемещается снизу вверх, сделав ряд, он перемещается влево и также идет в обратную сторону. Если садовник видит, что участок сада уже обработан другим садовником или является необрабатываемым, он идет дальше. Если по пути какой-то участок занят другим садовником, то садовник ожидает когда участок освободится, чтобы пройти дальше на доступный ему необрабатываемый участок. Садовники должны работать одновременно со скоростями, **определенными как параметры задачи.** Прохождение через любой квадрат занимает некоторое время, которое задается **константой**, меньшей чем времена обработки и принимается за единицу времени.

Создать клиент–серверное приложение, моделирующее работу садовников.

Каждого садовника представить отдельным клиентом. Сам сад — сервер.

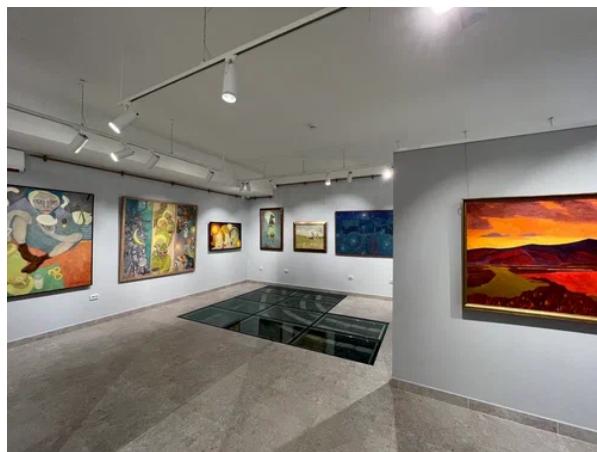
22. **Задача о картинной галерее.** Вахтер следит за тем, чтобы в картинной галерее одновременно было не более **25** посетителей. Для обозрения представлены **5** картин. Каждый посетитель **случайно** переходит от картины к картине, но если на желаемую картину любуются более пяти посетителей, он стоит в стороне и ждет, пока число желающих увидеть эту картину не станет меньше. По-



сетитель покидает галерею по завершении осмотра всех картин. Каждый посетитель уникальный, то есть имеет свой номер (например, PID). В галерею также пытаются постоянно зайти новые посетители, которые ожидают своей очереди и разрешения от вахтера, если та заполнена.

Создать клиент–серверное приложение, моделирующее однодневную работу картинной галереи (например, можно ограничить числом посетителей от 100 до 300).

Вахтер — сервер. Картичная галерея — клиент. Приходящие посетители реализуются общим клиентом. Время нахождения возле картины для каждого посетителя является случайной величиной в некотором диапазоне.



23. **Задача о болтунах.** **N** болтунов имеют телефоны. Они либо неко-

торое (**случайное**) время ждут звонков, либо звонят друг другу, чтобы побеседовать. Если телефон **случайного** абонента занят, болтун будет звонить другому **случайному** абоненту, пока ему кто-нибудь не ответит. Побеседовав **некоторое** время, болтун или ждет звонка, или звонит на другой случайный номер.

Создать клиент–серверное приложение, моделирующее поведение болтунов.

Каждый болтун – отдельный клиент. Сервер получает число болтунов при запуске и является коммутатором. Для автоматизации процесса запуск клиентов можно осуществлять скриптом. Также можно каждого клиента запускать вручную.



24. **Задача о программахистах.** В отделе работают три программиста. Каждый программист пишет свою программу и отдает ее на проверку одному из двух оставшихся программистов, выбирая его случайно и ожидая окончания проверки. Программист начинает проверять чужую программу, когда его собственная уже написана и передана на проверку. По завершении проверки, программист возвращает программу с результатом (формируемым случайно по

любому из выбранных Вами законов): программа написана правильно или неправильно. Программист «спит», если отправил свою программу и не проверяет чужую программу. Программист «просыпается», когда получает заключение от другого программиста. Если программа признана правильной, программист пишет другую программу, если программа признана неправильной, программист исправляет ее и отправляет на проверку тому же программисту, который ее проверял. К исправлению своей программы он приступает, завершив проверку чужой программы. При наличии в очереди проверяемых программ и приходе заключения о неправильной своей программы программист может выбирать любую из возможных работ. Необходимо учесть, что кто-то из программистов может получать несколько программ на проверку.

Создать клиент–серверное приложение, моделирующее работу программистов.

Каждый программист – отдельный клиент. Сервер обеспечивает передачу данных между клиентами.



25. **Задача об инвентаризации по рядам.** После нового года в библиотеке университета обнаружилась пропажа каталога. После поиска и наказания виноватых, ректор дал указание восстановить

каталог силами студентов. Фонд библиотека представляет собой прямоугольное помещение, в котором находится **M** рядов по **N** шкафов по **K** книг в каждом шкафу.

Требуется создать клиент–серверное приложение, составляющее каталог по информации, поступающей от каждого из студентов, проверяющих местоположение книг.

Сервер формирует каталог, получая информацию от студентов — клиентов. Каждый студент передает серверу информацию об одном ряде, после чего получает от сервера информацию о следующем ряде, который нужно обработать. Порождение студентов может быть вручную или с использованием скрипта.

Следует помнить, что в данном случае при занесении данных в уже заполняемый каталог придется производить сортировку своих данных с уже внесенными.

Важное примечание. Каталог — это список книг, упорядоченный (отсортированный) по названию книги (в данном случае в качестве названия можно взять и целое число — не обязательно использовать ASCII строку символов). Каждая строка каталога содержит: название книги (номер или строку), местоположение книги, включающее номер ряда, номер шкафа, номер книги в шкафу. Перед запуском процессов по составлению каталогов необходимо случайным образом расположить книги в шкафах, используя генератор случайных чисел. Также нужно предварительно вывести список размещения книг по шкафам, чтобы в конце сопоставить его с тем, как они оказались описанными в каталоге.

26. **Задача об инвентаризации по книгам.** После нового года в библиотеке университета обнаружилась пропажа каталога. После поиска и наказания, виноватых ректор дал указание восстановить каталог силами студентов. Фонд библиотека представляет собой прямоугольное помещение, в котором находится **M** рядов по **N** шкафов по **K** книг в каждом шкафу.

Требуется создать многопроцессное приложение, составляющее каталог по информации, поступающей от каждого из студентов, проверяющих местоположение книг.

При решении задачи использовать метод «портфель задач». Сервер осуществляет систематизацию каталога, получая информацию от нескольких клиентов-студентов. Каждый клиент передает информацию серверу о каждой отдельной книге и получает информацию о том, на каком месте получить информацию о следующей книге. То есть каждый студент регистрирует в каталоге по отдельности каждую книгу, после чего снова идет к шкафам за информацией о следующей книге.

Следует помнить, что в данном случае при занесении данных в уже заполняемый каталог придется производить сортировку своих данных с уже внесенными.



Важное примечание. Каталог — это список книг, упорядоченный (отсортированный) по названию книги (в данном случае в качестве названия можно взять и целое число — не обязательно использовать ASCII строку символов). Каждая строка каталога содержит: название книги (номер или строку), местоположение книги, включающее номер ряда, номер шкафа, номер книги в шкафу. Перед запуском процессов по составлению каталогов необходимо случайным образом расположить книги в шкафах, используя генератор случайных чисел. Также нужно предварительно вывести список размещения книг по шкафам, чтобы в конце сопоставить его с тем, как они оказались описаными в каталоге.

27. **Задача о привлекательной студентке.** У одной очень привлекательной студентки есть N поклонников. Традиционно в день

св. Валентина очень привлекательная студентка проводит романтический вечер с одним из поклонников. Счастливый избранник заранее не известен. С утра очень привлекательная студентка получает N «валентинок» с различными вариантами романтического вечера. Выбрав наиболее заманчивое предложение, студентка извещает счастливчика о своем согласии, а остальных — об отказе.

Требуется создать клиент–серверное приложение, моделирующее поведение студентки.

Каждый из студентов должен быть представлен отдельным клиентом. Студентка — сервер.

При решении использовать парадигму «клиент–сервер» с активным ожиданием (предполагается, что поклонник не сразу получает ответ, а только после того, как все они сделают запросы, после чего студентка выберет лучшего и разошлет все соответствующие уведомления). В протоколе вывода отметить реакцию каждого из студентов на информацию, полученную от студентки...



28. **Задача о производстве булавок.** В цехе по заточке булавок все необходимые операции осуществляются рабочими на трех производственных участках. На первом участке $1 \leq K \leq 3$ рабочих получают тупые булавки и в течение некоторого случайного времени каждый из них проверяет ее на предмет кривизны. Если булавка не кривая, то рабочий передает ее на второй участок, на котором $1 \leq L \leq 5$ работников осуществляют заточку. Булавка попадает

случайно выбранному из них свободному работнику, который осуществляет заточку и передает заточенную булавку на третий участок, на котором $1 \leq M \leq 2$ работников осуществляют контроль качества операции.

Требуется создать клиент–серверное приложение, моделирующее работу цеха.

Каждый работник – это отдельный клиент. Сервер осуществляет передачу булавок между соответствующей парой рабочих.

При решении использовать парадигму «производитель–потребитель». Следует учесть, что каждая из операций выполняется за случайное время, которое не связано с конкретным рабочим. Передача булавок осуществляется непосредственно от одного рабочего другим. То есть, если все рабочие на определенном участке заняты, то осуществляется ожидание одного из тех, кто освободится.



29. **Задача про экзамен.** Преподаватель проводит экзамен у группы студентов. Каждый студент получает свой билет, сообщает его номер и готовит письменный ответ за некоторое случайное время. Подготовив ответ, он передает его преподавателю. Преподаватель некоторое случайное время просматривает ответ и сообщает студенту оценку.

Требуется создать клиент–серверное приложение, моделирующее действия преподавателя и студентов.

Преподаватель — сервер. Каждый студент — отдельный клиент. Для автоматизации запуска студентов можно использовать скрипт, организующий поступление клиентов со случайной задержкой. Необходимо учесть, что одновременно сдавать экзамен может несколько студентов. То есть, сервер должен обслуживать более одного клиента.



30. **Военная операция.** Темной–темной ночью прапорщики Иванов, Петров и Нечепорук занимаются хищением военного имущества со склада родной военной части. Будучи умными людьми и отличниками боевой и строевой подготовки, прапорщики ввели разделение труда. Иванов многократно выносит имущество со склада и передает его в руки Петрову, который грузит его в грузовик. Нечепорук стоит на шухере и заодно подсчитывает рыночную стоимость добычи поле погрузки в грузовик очередной партии похищенного.

Требуется разработать клиент–серверное приложение, моделирующее деятельность прапорщиков.

Каждый прапорщик — отдельный клиент. Сервер используется для передачи информации о продвижении имущества между прапорщиками.

Необходимо учесть случайное время выполнения каждым прапорщиком своей боевой задачи и организовать в программе корректную их синхронизацию.



31. **Задача о Пути Кулака.** На седых склонах Гималаев стоит древний буддистский монастырь: Гуань-Инь-Янь. Каждый год в день сошествия на землю боддисатвы монахи монастыря собираются на совместное празднество и показывают свое совершенствование на Пути Кулака. Всех соревнующихся монахов первоначально разбивают на пары. Бои продолжаются до выявления победителя. Монах который победил в финальном бою, забирает себе на хранение статую боддисатвы.

Реализовать клиент–серверное приложение, определяющее победителя.

Каждый монах – отдельный клиент. Сервер используется для распределения пар и формирования результатов поединка. Програвший монах–клиент отключается. Победивший – продвигается дальше с новой энергией.

В качестве входных данных используется начальное значение в каждом клиенте, в котором хранится количество энергии Ци каждого монаха. При победе монах забирает энергию Ци своего противника. Новые пары образуются среди победителей других пар в порядке завершения поединков. То есть, возможна ситуация, когда бойцы, участвующие в поединке могут быстро победить и начать биться с другими, в то время как поединки начавшиеся ранее, могут продолжаться. Причем длительное время. Каждый поединок протекает некоторое случайное время, которое пропорционально

отношению энергии Ци побежденного к энергии Ци победителя, умноженному на поправочный коэффициент, позволяющий отслеживать протекание поединка на экране дисплея (например, путем умножения этого отношения на 1000 миллисекунд или другое более удобное значение).



32. Задача об Острове Сокровищ. Шайка пиратов под предводительством Джона Сильвера высадилась на берег Острова Сокровищ. Не смотря на добытую карту острова старого Флинта, местоположение сокровищ по-прежнему остается загадкой, поэтому искать клад приходится практически на ощупь. Так как Сильвер ходит на деревянной ноге, то самому бродить по джунглям ему не с руки. Джон Сильвер поделил остров на участки, а пиратов на небольшие группы. Каждой группе поручается искать клад на одном из участков, а сам Сильвер ждет на берегу. Пираты, обшарив свой участок, возвращаются к Сильверу и докладывают о результатах.

Требуется создать клиент–серверное приложение, моделирующее действия Сильвера и пиратов.

Сильвер — сервер, группы пиратов — клиенты

Примечание. Количество участков превышает число поисковых групп.

33. Пляшущие человечки. На тайном собрании глав преступного мира города Лондона председатель собрания профессор Мориарти



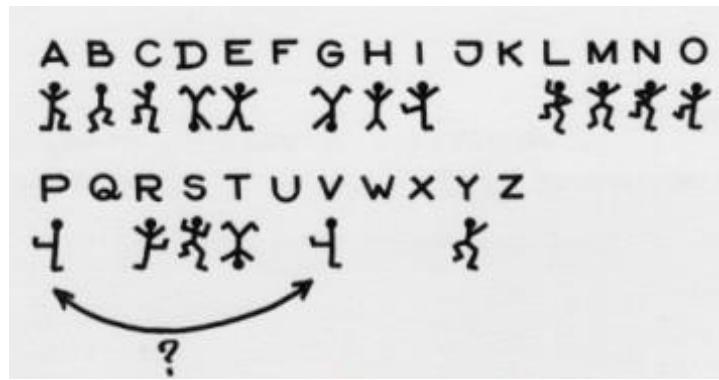
постановил: отныне вся переписка между преступниками должна вестись тайнописью. В качестве стандарта были выбраны «пляшущие человечки», шифр, в котором каждой букве латинского алфавита соответствует хитроумный значок.

Реализовать клиент–серверное приложение, шифрующее исходный текст (в качестве ключа используется кодовая таблица, устанавливающая однозначное соответствие между каждой буквой и каким–нибудь числом).

Каждый процесс–шифровальщик является клиентом. Он кодирует свои кусочки общего текста. При решении использовать paradigmу портфеля задач. клиенты работают асинхронно, формируя свой закодированный фрагмент в случайное время. Следовательно, при занесении информации в портфель–сервер необходимо проводить упорядочение фрагментов.

В программе необходимо вывести исходный текст, закодированные фрагменты по мере их формирования, окончательный закодированный текст.

34. **Задача о сельской библиотеке.** В библиотеке имеется N книг, Каждая из книг в одном экземпляре. M читателей регулярно заглядывают в библиотеку, выбирая для чтения от одной до трех книг и читая их некоторое количество дней. Если желаемой книги нет, то читатель, взяв существующие, дожидается от библиотекаря информации об ее появлении и приходит в библиотеку, чтобы



специально забрать ее. Возможна ситуация, когда несколько читателей конкурируют из-за этой популярной книги.

Создать клиент–серверное приложение, моделирующее заданный процесс.

Библиотекарь — сервер. Читатели — отдельные клиенты.



35. «Камень, ножницы, бумага» 1 — однокруговая система. N студентов, изнывающих от скуки на лекции по операционным системам решили организовать однокруговой турнир в игру «Камень, ножницы, бумага». Каждый с каждым при этом играет только один раз. За победу дается два очка, за ничью — одно, за поражение — ноль.

Требуется создать клиент–серверное приложение, моделирующее турнир.

Каждый студент — отдельный клиент. Генерация камня, ножниц и бумаги в каждом поединке формируется сервером случайно. Завершение работы программы осуществляется по завершению

турнира. Количество участвующих студентов задается сервером.



36. «Камень, ножницы, бумага» 2 — олимпийская система.
 N студентов, изнывающих от скуки на лекции по операционным системам решили организовать турнир в игру «Камень, ножницы, бумага» по олимпийской системе (с выбыванием). В случае ничей (выпадение одинаковых предметов) игра продолжается до победы одного из участников.

Требуется создать клиент–серверное приложение, моделирующее турнир.

Каждый студент — отдельный клиент. Генерация камня, ножниц и бумаги в каждом поединке формируется сервером случайно. Завершение работы программы осуществляется по завершению турнира. Количество участвующих студентов задается сервером.



Требование к выполнению и оценке

Для выполнения задания на более высокую оценку необходимо разработать программу **для каждой нынешней оценки**. Рекомендуется каждую программу размещать в отдельном каталоге. Демонстрация результатов работы программы возможна с использованием любых вариантов. Например с использованием файлов с результатами работы, с применением скриншотов или видео.

Так как большинство задач связаны с моделированием различных ситуаций, необходимо обеспечить наглядное представление результатов. Для интерактивного отображения на дисплее рекомендуется использовать временные задержки. Рекомендуется использовать аргументы командной строки для запуска заданного числа процессов. Также для параллельного запуска множества процессов или клиентов допускается использование соответствующих скриптов, в котором программы запускаются в фоновом режиме. В целом все определяется условием задачи и фантазией разработчика программы.

Примечание. В представленном задании в ряде случаев оценка за каждый пункт задается диапазоном. То есть возможно ее понижение за счет того, что некоторые требования будут при проверке считаться невыполнеными.

Программы должны быть написаны на языке программирования C/C++ и выполняться в среде ОС Linux. При этом для организации сетевых взаимодействий должны использоваться функции операционной системы, изучаемые в рамках дисциплины.

При отсутствии корректной реализации задача считается нерешиенной (0 баллов).

4–5 баллов

Разработать клиент–серверное приложение, в котором сервер (или серверы) и клиенты независимо друг от друга отображают только ту информацию, которая поступает им во время обмена. То есть, отсутствует какой-либо общий вывод интегрированной информации, отображающий поведение системы в целом.

Требования:

1. В отчете необходимо привести фамилию, имя, отчество исполнителя, группу.
2. Привести номер варианта и условие задачи.
3. Представить сценарий решаемой задачи поясняющий, каким образом исходные сущности и их поведение отображаются в серверы, клиенты, процессы и как осуществляется их взаимодействие.
4. При запуске программ требуемые для их работы IP адреса и порты необходимо задавать в командной строке, чтобы обеспечить гибкую подстройку к любой сети.
5. Для обеспечения корректного взаимодействия сетевых приложений и существующих в них процессов допускается использовать любые ранее изученные программные объекты.
6. Разработанное приложение должно работать как на одном компьютере так и в распределенном (сетевом) режиме на нескольких компьютерах, по которым можно будет разнести серверы и клиентов.
7. Результаты работы программы должны быть отражены в отчете.
8. Завершение работы клиентов и серверов на данном этапе не оговаривается. Но оно должно быть представлено в сценарии.

6–7 баллов

В дополнение к программе на предыдущую оценку необходимо разработать отдельную клиентскую программу, подключаемую к серверу, которая предназначена для отображение комплексной информации о выполнении приложения в целом. То есть, данный программный модуль должен адекватно и в полном виде отображать поведение моделируемой системы (информацию передаваемую на сервер и информацию, порождаемую сервером), позволяя не пользоваться отдельными видами, предоставляемыми клиентами и серверами по отдельности.

Отчет расширить информацией о добавленном клиенте, модификациях других частей программы. Привести соответствующие результаты работы данной программы.

8 баллов

В дополнение к предыдущей программе реализовать возможность, подключения множества клиентов, обеспечивающих отображение информации о работе приложения. Это должно позволить осуществлять наблюдение за поведением программы с многих независимых компьютеров. При этом клиентов–наблюдателей можно отключать и подключать снова в динамическом режиме без нарушения работы всего приложения.

Отчет расширить информацией о добавленной реализации и привести соответствующие результаты работы программы.

9 баллов

В дополнение к программам на предыдущие оценки необходимо разработать приложение, позволяющее отключать и подключать различных клиентов с сохранением работоспособности сервера. После этого можно вновь запускать отключенных клиентов, чтобы приложение в целом могло продолжить свою работу.

Отчет расширить информацией о добавленной реализации и привести соответствующие результаты работы программы.

10 баллов

Расширить предыдущую программу таким образом, чтобы при завершении работы сервера происходило корректное завершение работы всех подключенных клиентов. То есть, данная программа должна являться модификацией программы на оценку в 9 баллов. Отдельную программу на 9 баллов в этом случае сдавать не нужно

Отчет расширить информацией о добавленной реализации и привести соответствующие результаты работы программы.