

27. **Задача про экзамен.** Преподаватель проводит экзамен у группы студентов. Каждый студент получает свой билет, сообщает его номер и готовит письменный ответ. Подготовив ответ, он передает его преподавателю. Преподаватель просматривает ответ и сообщает студенту оценку. Студент, дождавшись результата, уходит с экзамена. *Требуется создать приложение, моделирующее действия преподавателя и студентов, каждый из которых представлен отдельным процессом. Преподаватель — сервер. Каждый студент — отдельный клиент.*

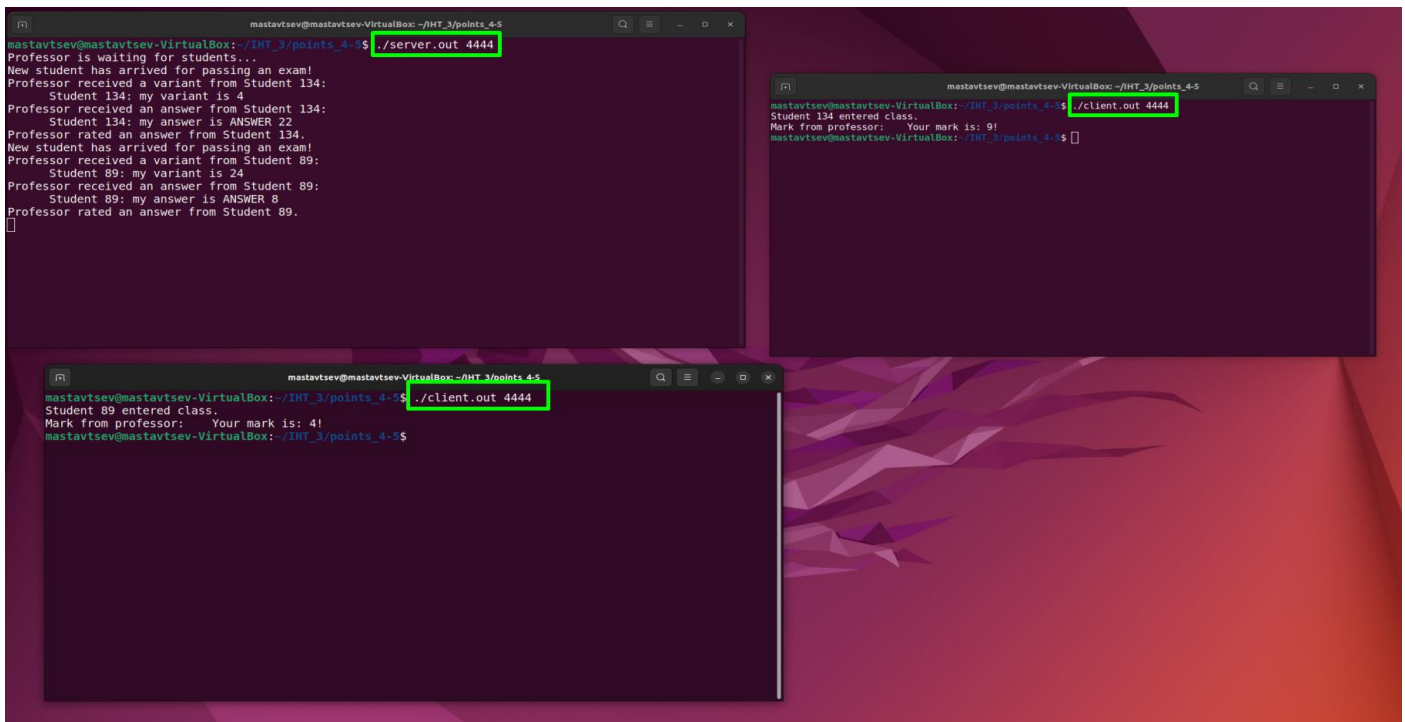
#### 4 - 5 баллов (именованные POSIX семафоры)

##### Файлы и каталоги:

*server.c*

*client.c*

##### Ввод и вывод данных (сервер и два клиента):



```
mastavtsev@mastavtsev-VirtualBox: ~/HT_3/points_4-5
mastavtsev@mastavtsev-VirtualBox:~/HT_3/points_4-5$ ./server.out 4444
Professor is waiting for students...
New student has arrived for passing an exam!
Professor received a variant from Student 134:
  Student 134: my variant is 4
Professor received an answer from Student 134:
  Student 134: my answer is ANSWER 22
Professor rated an answer from Student 134.
New student has arrived for passing an exam!
Professor received a variant from Student 89:
  Student 89: my variant is 24
Professor received an answer from Student 89:
  Student 89: my answer is ANSWER 8
Professor rated an answer from Student 89.

```

```
mastavtsev@mastavtsev-VirtualBox:~/HT_3/points_4-5$ ./client.out 4444
Student 134 entered class.
Mark from professor: Your mark is: 9!
mastavtsev@mastavtsev-VirtualBox:~/HT_3/points_4-5$

```

```
mastavtsev@mastavtsev-VirtualBox:~/HT_3/points_4-5$ ./client.out 4444
Student 89 entered class.
Mark from professor: Your mark is: 4!
mastavtsev@mastavtsev-VirtualBox:~/HT_3/points_4-5$

```

Параметрами для исполняемых файлов являются:

server.c – порт сервера, на котором он прослушивает клиентов (4444)

client.c – тот же номер порта, на который посылаются сообщения (4444)

Важно отметить, что здесь и далее параметрами не будет являться значение IP, так как в виртуальной среде Virtual Box при указании конкретного значения IP у меня вылетала ошибка Permission Denied, то есть система почему-то отказывала в доступе к localhost. Поэтому везде при конфигурации IP сокета указывался флаг INADDR\_ANY.

### **Процесс сдачи экзамены (выходные данные):**

Данный раздел актуален для всех программ, поэтому пояснения будут только тут.

Процесс сдачи экзамен происходит следующим образом:

1. Студент тянет билет -> сообщает номер варианта профессору (через канал/способ передачи информации, обозначенный в задании на конкретный балл).
2. Профессор получает сообщение от студента, выводит его и продолжает «принимать экзамен» - ожидает сообщения от других студентов.
3. После сообщения варианта студент «решает свой билет» и сообщает ответ преподавателю, так же необходимым в задаче способом (ответом является просто фраза ANSWER “random num”). Далее он ждёт, когда профессор сообщит ему оценку.
4. Профессор думает над оценкой студента (sleep(1)) и сообщает ему.
5. Студент, получив оценку от профессора, завершает работу.

Передача сообщений между процессами сервера (профессор) и клиентов (студенты) происходят за счёт протокола **TCP**.

К серверу могут подключаться одновременно сразу несколько клиентов. Ровно как преподаватель принимает экзамен сразу у группы студентов, сервер работает сразу с группой клиентов. То есть обработка каждого нового поступившего клиента ведётся в дочернем процессе.

Передача сообщений от студента к профессору происходит на основе структуры данных Message:

student\_id - уникальный номер студента, используется для различия студентов

message\_type - тип сообщения:

message\_type == 0 - сообщение о номере варианта

message\_type == 1 - ответ на вариант

message - само сообщение

## 6-7 баллов

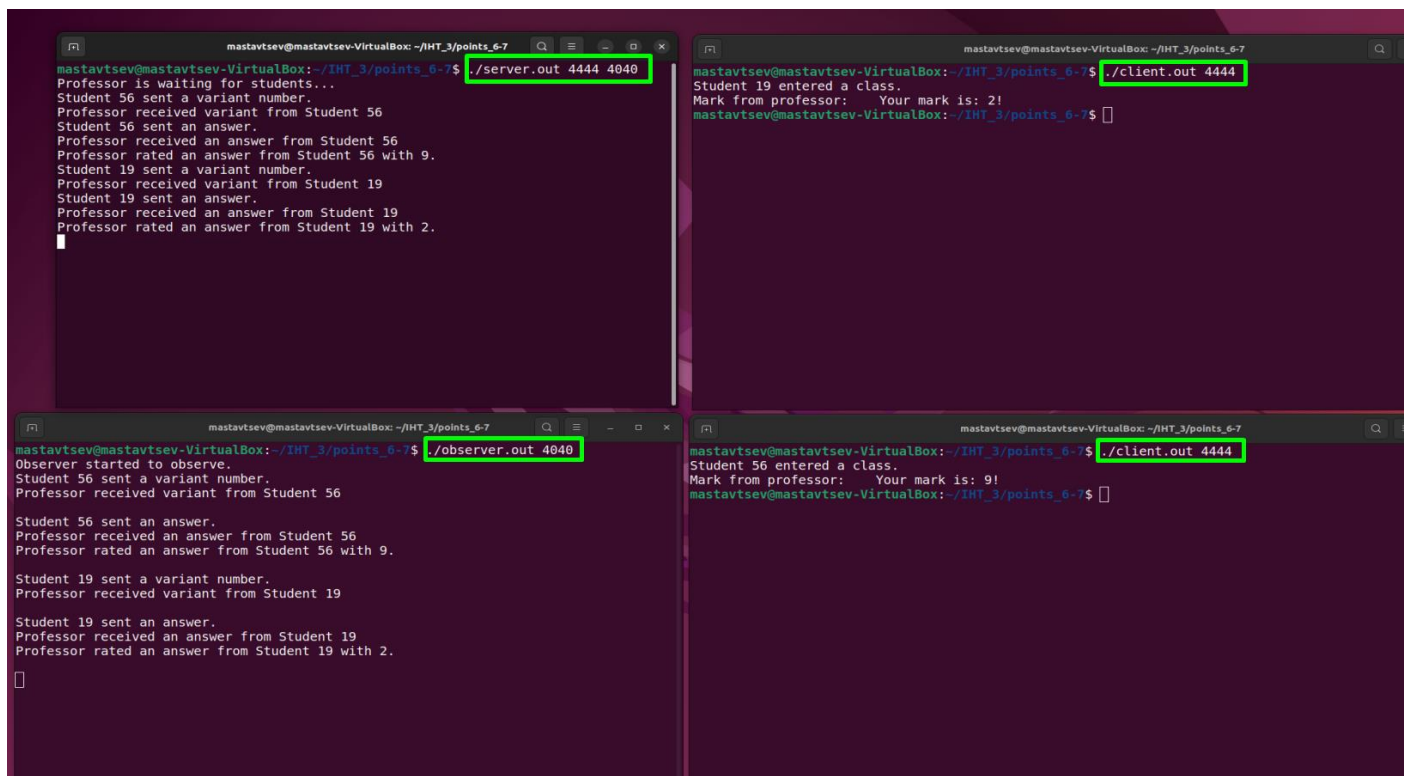
### Файлы и каталоги:

*server.c*

*client.c*

*observer.c*

### Ввод и вывод данных (сервер, два клиента и единственный наблюдатель):



The image displays four terminal windows arranged in a 2x2 grid, showing the execution of a program. Each window has a title bar indicating the user is 'mastavtsev' on a 'mastavtsev-VirtualBox' machine, with the current directory being '~/IHT\_3/points\_6-7'.

- Top-left window:** The prompt is `./server.out 4444 4040`. The output shows a professor waiting for students, receiving variant numbers and answers from students 56 and 19, and rating their answers (9 and 2 respectively).
- Top-right window:** The prompt is `./client.out 4444`. The output shows student 19 entering a class and receiving a mark of 2 from the professor.
- Bottom-left window:** The prompt is `./observer.out 4040`. The output shows an observer starting to observe, receiving variant numbers and answers from students 56 and 19, and rating their answers (9 and 2 respectively).
- Bottom-right window:** The prompt is `./client.out 4444`. The output shows student 56 entering a class and receiving a mark of 9 from the professor.

Параметрами для исполняемых файлов являются:

server.c – порт сервера, на котором он прослушивает клиентов (4444) и порт сервера, на котором он устанавливает соединение с наблюдателем (4040)

client.c – номер порта, на который посылаются сообщения от сервера клиенту (4444)

observer.c - номер порта, на который посылаются сообщения от сервера наблюдателю (4040)

Важно отметить, что важен порядок запуска процессов различных агентов. Сначала необходимо запустить процесс сервера, далее запускается процесс наблюдателя, и уже после желаемое количество процессов клиентов.

### **Общая схема решаемой задачи:**

Отличием данной задачи является наличие ещё одного типа клиента сервера – наблюдателя. Для корректного взаимодействия с двумя различными типами клиентов на сервера создаётся два различных сокета, которые прослушивают два различных порта. В приложении сервера создается два процесса, один из которых осуществляет взаимодействие с клиентами-студентами, а другой процесс работает с клиентами-наблюдателями.

Также важным отличием является то, что информация о процессе взаимодействия сервера-профессора с клиентами-студентами должна передаваться другому клиенту-наблюдателю. Для реализации передачи информации между процессами в рамках приложения сервера была выбрана разделяемая память в стандарте POSIX, с использованием семафоров также в стиле POSIX для осуществления синхронизации между ними.

## 8 баллов

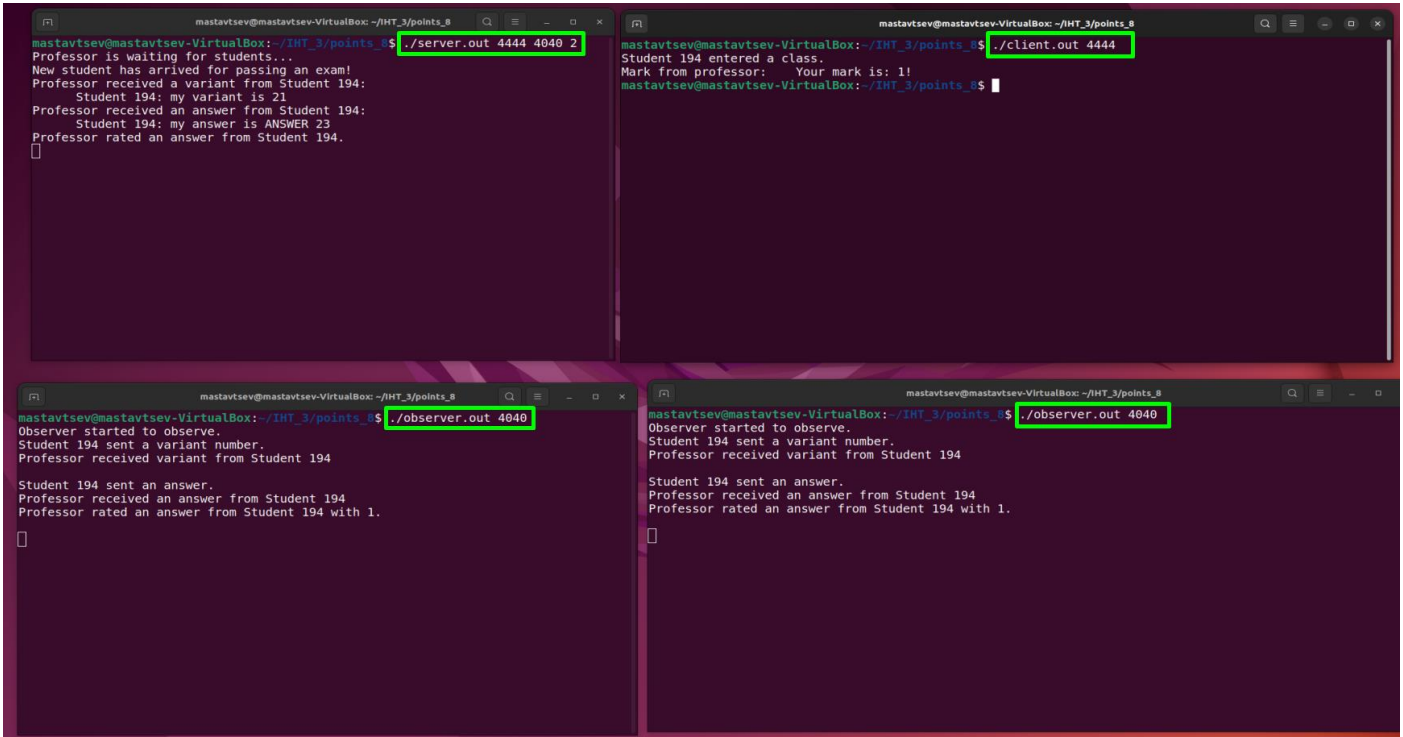
### Файлы и каталоги:

*server.c*

*client.c*

*observer.c*

### Ввод и вывод данных (сервер, один клиент и два наблюдателя):



The image displays four terminal windows arranged in a 2x2 grid, showing the execution of a distributed system simulation. Each window has a title bar indicating the user is 'mastavtsev' on a 'mastavtsev-VirtualBox' machine, with the current directory being '~/IHT\_3/points\_8'.

- Top-left window:** The command `./server.out 4444 4040 2` is executed. The output shows a professor waiting for students, a new student (194) arriving, and the professor receiving an answer from Student 194.
- Top-right window:** The command `./client.out 4444` is executed. The output shows Student 194 entering a class and receiving a mark of 1! from the professor.
- Bottom-left window:** The command `./observer.out 4040` is executed. The output shows the observer starting to observe, receiving a variant number from Student 194, and then receiving an answer from Student 194, which is rated with 1.
- Bottom-right window:** The command `./observer.out 4040` is executed. The output shows the observer starting to observe, receiving a variant number from Student 194, and then receiving an answer from Student 194, which is rated with 1.

Параметрами для исполняемого файла являются:

*server.c* – (1) порт сервера, на котором он прослушивает клиентов (4444) и (2) порт сервера, на котором он устанавливает соединение с наблюдателем (4040) и (3) количество наблюдателей  $N$

*client.c* – номер порта, на который посылаются сообщения от сервера клиенту (4444)

*observer.c* - номер порта, на который посылаются сообщения от сервера наблюдателю (4040)

Спецификой задачи на данный балл является необходимость осуществлять рассылку информации о взаимодействия сервера-профессора и клиента-студента сразу нескольким клиентам-наблюдателям.

Наилучшим решением для реализации данной задачи было бы выбрать в качестве способа взаимодействия с клиентами-наблюдателями протокол UDP, так как его особенностью и является возможность вести широковещательную рассылку.

Однако, так как необходимо использовать только инструменты протокола TCP используется схема, описанная далее.

Важно отметить, что важен порядок запуска процессов различных агентов. Сначала необходимо запустить процесс сервера, далее запускаются  $N$  процессов наблюдателей, и уже после желаемое количество процессов клиентов.

### **Общая схема решаемой задачи:**

Для передачи информации между процессами также используется разделяемая память и семафоры стандарта POSIX. Приложение сервера также создаёт два процесса для работы с двумя типами клиентов.

Отличием от задачи на 6-7 баллов является поддержка сразу множества клиентов-наблюдателей. Для её реализации в подпроцессе приложения клиента вызывается цикл *for*, который  $N$  устанавливает соединение с клиентами-наблюдателями и сохраняет их сокеты (файловые дескрипторы) в массив файловых дескрипторов.

Далее при получении информации о взаимодействии преподавателя и студентов также циклом *for* и использования массива сокетов каждому клиенту наблюдателю отправляется сообщение с информацией.

При работе над программой для данной оценки я также пытался реализовать подход, основанный на использовании дополнительного участка разделяемой памяти, в котором бы сохранял количество подключённых наблюдателей и их файловый дескрипторы. Однако, у меня возникала проблема с передачей значения сокетов (файловых

дескрипторов) между процессами, поэтому от данного подхода решено было уйти – его можно изучить в файле `server-alternative.c`