Студент Крамаренко М.К., группа 234, ИДЗ №3, вариант №30

### Задание:

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 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.

**Обращаю внимание**, что в каждой программе есть файл launch.sh, который отвечает за компиляцию программ, а также выводит инструкцию по запуску программы.

# **Реализация на 4-5 балла. Программа находится в папке 04-05.**

**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. **Представить сценарий решаемой задачи поясняющий, каким образом исходные сущности и их поведение отображаются в серверы, клиенты, процессы и как осуществляется их взаимодействие.**

Общий сценарий решения:

1. **Сервер.** Инициализирует tcp сокет и ожидает подключение 3 клиентов: Иванова, Петрова и Нечепорука. Если клиент подключился повторно (например, 2 Иванова), то первое подключение закрывается.
2. **Клиент.** Подключается по переданному адресу и порту. После подключения сразу сообщает свой id. Начинает работу при поступлении сообщению по сокету.
3. **Управление и синхронизация.** После подключения всех клиентов на сервере, сервер поочерёдно выполняет действия:
   1. посылает сообщение Иванову, что пора начинать работу и ждёт результат его работы.
   2. посылает сообщение Петрову и ждёт его ответ.
   3. посылает сообщение Нечепоруку и ждёт его ответ.
4. Завершение. Программа завершается в двух случаях:

Первый. Иванов всё украл, и его программа посылает сообщение серверу со специальным id **IVANOV\_DONE\_ID**. Это является сигналом серверу о завершении работы. После чего все программы завершаются.

Второй. Получение сигнала **SIGINT** на любой программе. При получении такого сигнала, программы завершаются не дожидаясь, пока Иванов всё украдёт.

1. **При запуске программ требуемые для их работы IP адреса и порты необходимо задавать в командной строке, чтобы обеспечить гибкую подстройку к любой сети.**

Для всех программ указываются адрес и порт подключения. Для программы Иванова дополнительным опциональным параметром является количество вещей для кражи, по умолчанию **10**.

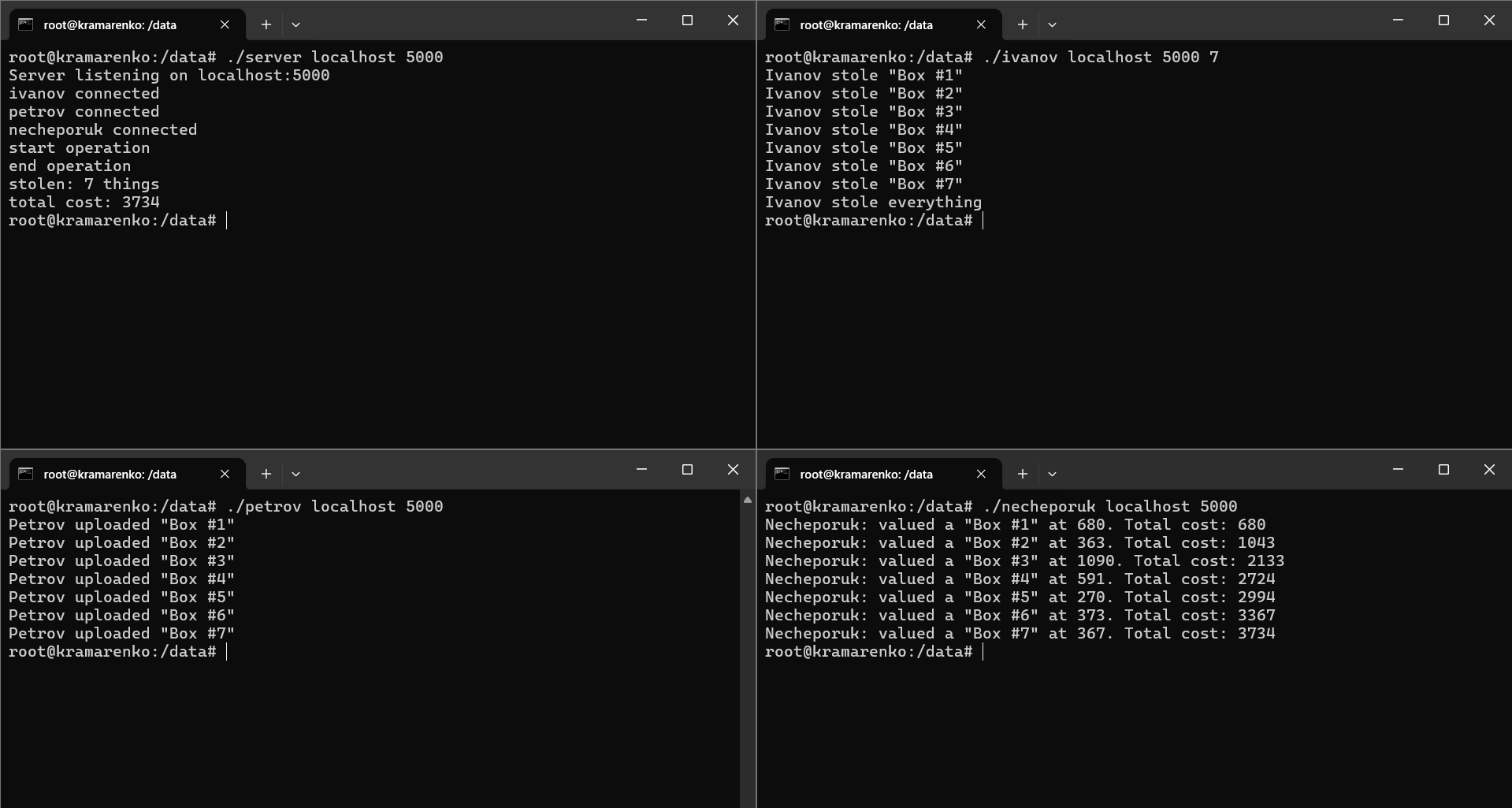
1. **Завершение работы клиентов и серверов на данном этапе не оговаривается. Но оно должно быть представлено в сценарии.**

Завершение. Программа завершается в двух случаях:

Первый. Иванов всё украл, и его программа посылает сообщение серверу со специальным id **IVANOV\_DONE\_ID**. Это является сигналом серверу о завершении работы. После чего все программы завершаются.

Второй. Получение сигнала **SIGINT** на любой программе. При получении такого сигнала, программы завершаются не дожидаясь, пока Иванов всё украдёт.

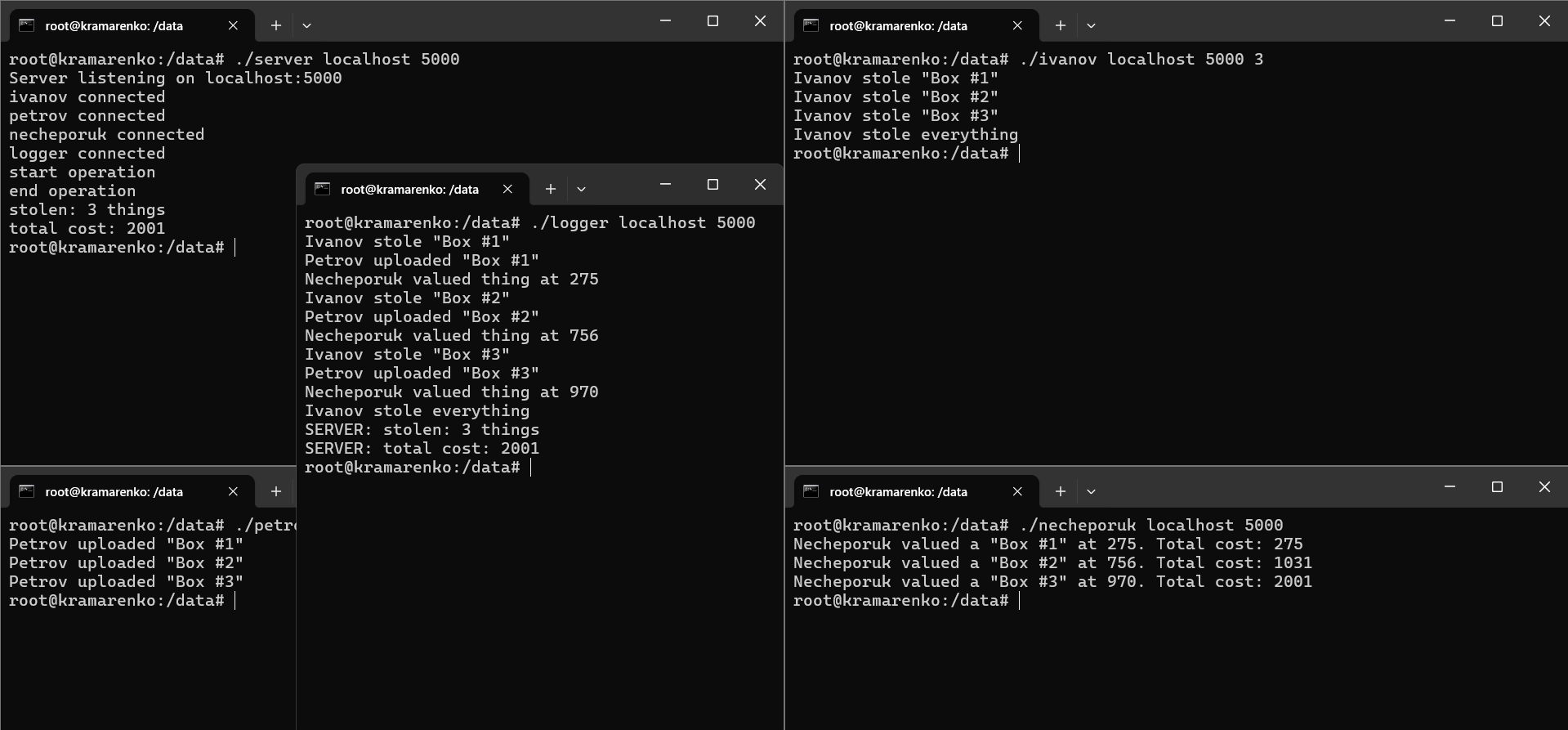
1. **Пример работы программы.**



# **Реализация на 6-7 баллов. Программа находится в папке 06-07.**

**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.**

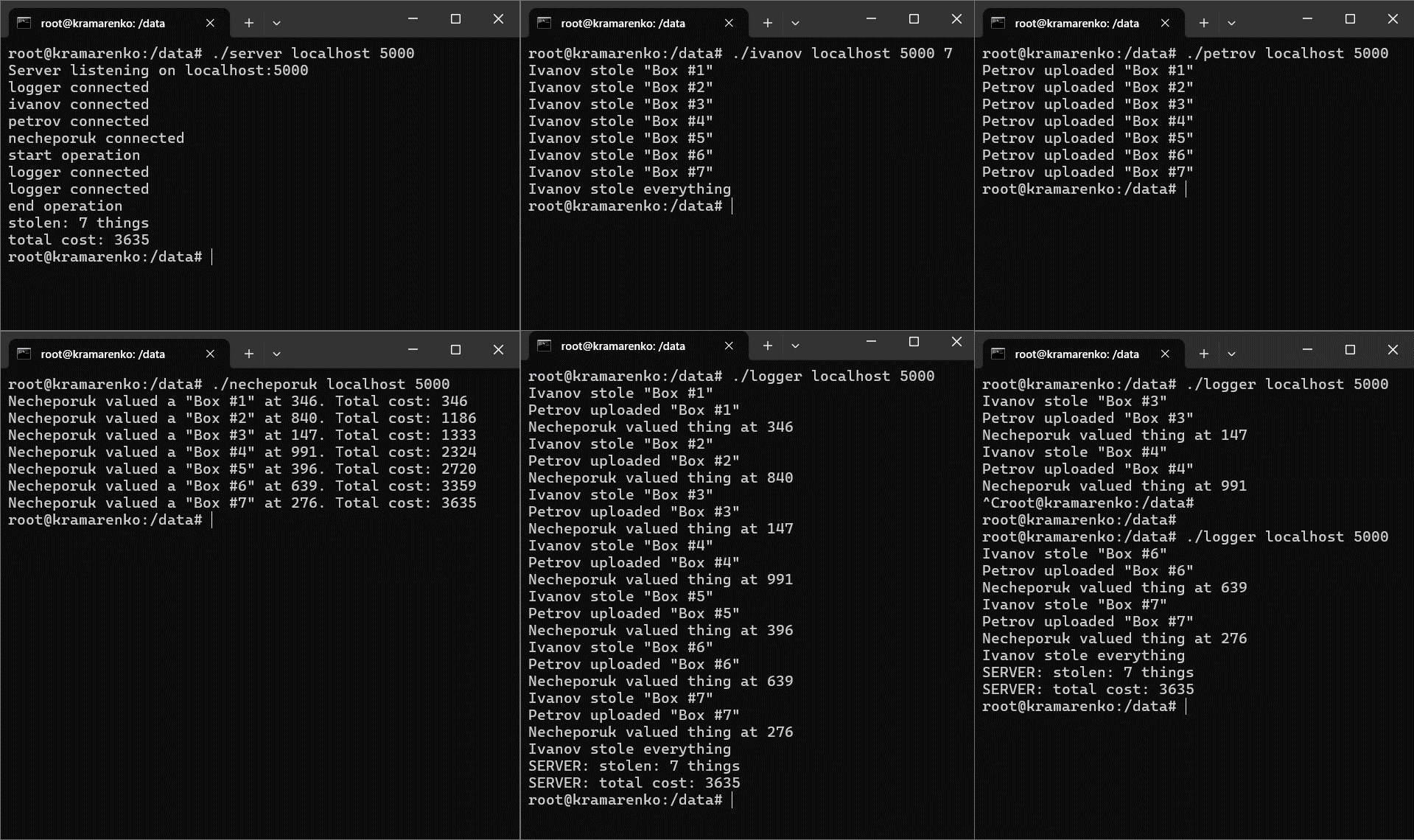
Была написана отдельная программа **logger.c**, которая подключается к серверу также, как и остальные программы. Сервер дублирует сообщения в **logger**, который выводит комплексную информацию о происходящем.



# **Реализация на 8 баллов. Программа находится в папке 08.**

**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.**

Пример работы приложения:



**6**

**логгер**

**5**

**логгер**

**4**

**Нечепорук**

**3**

**Петров**

**2**

**Иванов**

**1**

**сервер**

**Комментарий к логгерам.**

Логгер в окне №5 подключается в самом начале и наблюдает до конца работы приложения.

Логгер в окне №6 подключается позже начала работы программы и начинает выводить информацию с 3 итерации. Логгер наблюдает 2 итерации, после чего отключается. Затем заново подключается на 6 итерации и наблюдает до конца программы.

**Идея решения.**

Подключение клиентов остаётся прежним. Клиент логгера является опциональным, т.е. в начале программы логгера можно не подключать. Когда все необходимые клиенты запустились, устанавливается флаг **O\_NONBLOCK**, чтобы процесс программы при попытке подключить нового клиента не блокировался.

Итерация поделена на команды и перед выполнением каждой команды проверяется наличие новых подключений, если они отсутствуют, то программа переходит к выполнению команды.

# **Реализация на 9 баллов.**

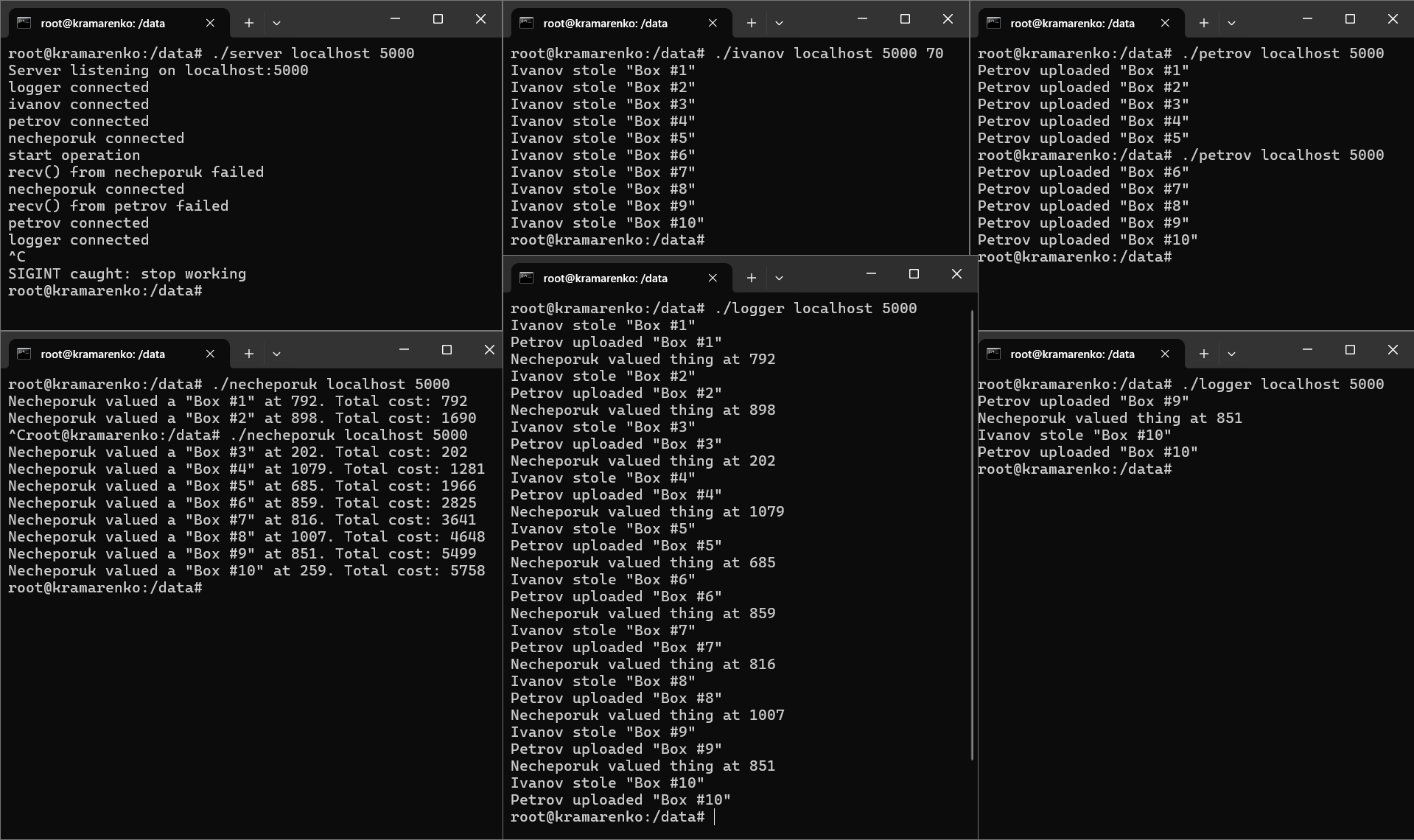
**В дополнение к программам на предыдущие оценки необходимо разработать приложение, позволяющее отключать и подключать различных клиентов с сохранением работоспособности сервера.**

# **Реализация на 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.**

**Программа находится в папке 09-10.**

Пример работы приложения:



**1**

**сервер**

[1] Тут было прерывание **Ctrl+C**

После Нечепорук подключился и

продолжил работу

[2] Тут было прерывание **Ctrl+C**

После Петров подключился и

продолжил работу

[2] Тут Петров пропал

И заново подключился

[1] Тут Нечепорук пропал

И заново подключился

**5**

**логгер**

**2**

**Иванов**

**3**

**Петров**

**4**

**Нечепорук**

**6**

**логгер**

Комментарий:

**Первое прерывание** клиента произошло на третьей итерации (окно №4). Информации, что клиент стал недоступен появилась на сервере (окно №1). Как только клиент вернулся, сервер отправил ему нужный запрос на обработку. Программа продолжила функционировать.

**Второе прерывание** клиента произошло на шестой итерации (окно №3). Информации, что клиент стал недоступен появилась на сервере (окно №2). Как только клиент вернулся, сервер отправил ему нужный запрос на обработку. Программа продолжила функционировать.

**Клиенты-логгеры** работают независимо, как в предыдущей программе.

**Третье прерывание** произошло на сервере (окно №1), в результате чего программа завершилась, клиенты также завершили свою работу.

Идея решения.

После инициализации сервера, сразу запускается главный цикла программы, который поочерёдно вызывает команды для различных клиентов. Но перед запуском этих команд производится попытка подключить нового клиента:

1. Сначала клиентов нет, поэтому вызов метода **accept()** является блокирующим. Так происходит, пока все необходимые клиенты не подключатся (Иванов, Петров, Нечепорук). Как только все необходимые клиенты подключились, устанавливается флаг **O\_NONBLOCK**, чтобы при повторном вызове **accept()** программа не ждала новых подключений. Когда все клиенты подключены, возвращается соответствующее значение, которое означается, что все клиенты готовы к работе.
2. Если все необходимые клиенты подключены, то перед каждым запуском команды всё равно проверяется **accept()** для того, чтобы подключить логгеров. Но если логгеров нет, то программа уже не ждёт нового подключения благодаря флагу **O\_NONBLOCK** и продолжает работу.
3. Сервер не может знать, когда именно пропал клиент. Он узнаёт об этом во время взаимодействия с клиентом. Если клиент не отвечает, то соответствующее подключение закрывается и флаг **O\_NONBLOCK** удаляется. И на новой итерации ожидается подключение необходимого клиента.