

Чанышева Диана Владимировна

БПИ237

Программная инженерия, 2 курс

**ИДЗ-4 по дисциплине  
«Операционные Системы»  
Вариант 12**

## Общие требования:

Цели и задачи:

Изучить работу с транспортным протоколом UDP. 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.

## Требования к варианту:

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.*

## Суть задачи, короче говоря, такая

По существу:

- СЕРВЕР

Центральный диспетчер очередей

Принимает от клиента-генератора UDP-сообщения о появлении новых покупателей

Хранит внутри себя две очереди по N мест

Решает, в какую очередь поставить покупателя, а если обе полны – отказывает

Отправляет по UDP сообщение соответствующему клиенту-кассиру о том, что к нему пришёл новый покупатель

Ну и может получать от кассира уведомление об окончании обслуживания, уменьшая длину очереди

- КЛИЕНТЫ

1. Клиент-генератор покупателей

Периодически шлет UDP-пакет на адрес сервера то, что появился новый покупатель

2. Два клиента-кассира

Каждый открывает свой UDP-порт и слушает сервер

Когда приходит просьба обслужить покупателя, моделирует время обслуживания (задержка), затем (например) отправляет серверу ответ «готов что делать дальше».

## Требования на выполнение задачи на 4-5 баллов:

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. (см выше, Суть задачи)**

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

Всё так, только вместо командной строки я использую bash-скрипт для запуска:



```
#!/bin/bash

gcc -o server server.c
gcc -o generator generator.c
gcc -o cashier cashier.c

echo "Запуск сервера..."
./server 9000 127.0.0.1 9001 127.0.0.1 9002 5 &
SERVER_PID=$!

echo "Запуск кассиров..."
./cashier 9001 127.0.0.1 9000 2000 &
CASHIER1_PID=$!
./cashier 9002 127.0.0.1 9000 3000 &
CASHIER2_PID=$!

echo "Запуск генератора покупателей..."
./generator 127.0.0.1 9000 1000 &
GENERATOR_PID=$!

echo "Все процессы запущены. Для остановки нажмите Ctrl+C"

trap 'kill $SERVER_PID $CASHIER1_PID $CASHIER2_PID $GENERATOR_PID; echo "Процессы остановлены"; exit' SIGINT

wait
```

Порты задаются в качестве аргументов, всё верно.

**5. Для обеспечения корректного взаимодействия сетевых приложений и существующих в них процессов допускается использовать любые ранее изученные программные объекты.**

Всё так

**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.**

Работает, проверено научным путём

**7. Результаты работы программы должны быть отражены в отчете.**

Результаты работы программы:

```
-bash: kill: (662) - No such process
diana@LAPTOP-NJFMCILS:/mnt/d/Документы Важные ДЗ/ОСИ/ОСИ_ИДЗ/ОСИ_ИДЗ4$ ps aux | grep -E 'server|cashier|generator'
root      7  0.0  0.0   2476   196 ?        S1   22:58   0:00 plan9 --control-socket 6 --log-level 4 --server-fd
--pipe-fd 9 --log-truncate
diana    668  0.0  0.0   4100  1924 pts/0    S+   23:13   0:00 grep --color=auto -E server|cashier|generator
diana@LAPTOP-NJFMCILS:/mnt/d/Документы Важные ДЗ/ОСИ/ОСИ_ИДЗ/ОСИ_ИДЗ4$ ./build.sh
Запуск сервера...
Запуск кассиров...
Запуск генератора покупателей...
Все процессы запущены. Для остановки нажмите Ctrl+C
[Cashier:9001] Waiting on port 9001, service time=2000 ms
[Cashier:9002] Waiting on port 9002, service time=3000 ms
[Server] Listening on port 9000, queuesize=5
[Generator] Sending new customers every 1000 ms to 127.0.0.1:9000
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=1).
[Cashier:9001] Serving customer...
[Generator] New customer sent.
[Server] Customer assigned to cashier 2 (queue=1).
[Cashier:9002] Serving customer...
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=2).
[Cashier:9001] Done.
[Server] Cashier 1 done. Remaining queue=1.
[Cashier:9001] Serving customer...
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=2).
[Cashier:9002] Done.
[Server] Cashier 2 done. Remaining queue=0.
[Generator] New customer sent.
[Server] Customer assigned to cashier 2 (queue=1).
[Cashier:9002] Serving customer...
[Cashier:9001] Done.
[Server] Cashier 1 done. Remaining queue=1.
[Cashier:9001] Serving customer...
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=2).
^CПроцессы остановлены
diana@LAPTOP-NJFMCILS:/mnt/d/Документы Важные ДЗ/ОСИ/ОСИ_ИДЗ/ОСИ_ИДЗ4$
```

Всё корректно отображается, программа завершает работу по ctrl+C

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

Я думала завершать работу по времени и сделать таймер, но с таймером почему-то моя программа не остановилась, и я решила, что это костыль, и нужно реально завершать работу по ctrl+C, реализовала это через баш-билд скрипт, а не программно (скрипт указан выше)

## Требования на выполнение задачи на 6-7 баллов:

### Задание:

**В дополнение к программе на предыдущую оценку необходимо разработать клиентскую программу, подключаемую к серверу, которая предназначена для отображение комплексной информации о выполнении приложения в целом. То есть, данный программный модуль должен адекватно отображать поведение моделируемой системы, 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.**

То есть теперь мы должны добавить некую клиентскую программу.

Клиентская программа будет получать информацию от сервера, кассиров и генератора, отображать данные о состоянии очередей и кассиров, а также выводить события, такие как новый покупатель, завершение обслуживания и т.д.

- 1) Сервер продолжает выполнять свою роль, распределяя клиентов по очередям
- 2) Кассиры обрабатывают клиентов и отправляют данные серверу
- 3) Генератор продолжает посылать новых клиентов серверу
- 4) Клиентский модуль для отображения информации получает данные от сервера и отображает их в удобном виде

Пусть клиент будет у нас в программном файле client.c

Тогда доработаем файл для запуска:

```
build - Блокнот
Файл  Правка  Формат  Вид  Справка
#!/bin/bash

gcc -o server server.c
gcc -o generator generator.c
gcc -o cashier cashier.c
gcc -o client client.c # Компиляция программы для отображения информации о системе

echo "Запуск сервера..."
./server 9000 127.0.0.1 9001 127.0.0.1 9002 5 &
SERVER_PID=$!

echo "Запуск кассиров..."
./cashier 9001 127.0.0.1 9000 2000 &
CASHIER1_PID=$!
./cashier 9002 127.0.0.1 9000 3000 &
CASHIER2_PID=$!

echo "Запуск генератора покупателей..."
./generator 127.0.0.1 9000 1000 &
GENERATOR_PID=$!

echo "Запуск клиента для отображения информации..."
./client 127.0.0.1 9000 &
CLIENT_PID=$!

echo "Все процессы запущены. Для остановки нажмите Ctrl+C"
trap 'kill $SERVER_PID $CASHIER1_PID $CASHIER2_PID $GENERATOR_PID $CLIENT_PID; echo "Процессы остановлены"; exit'
wait
```

И запустим:

```
[Cashier:9001] Waiting on port 9001, service time=2000 ms
[Generator] Sending new customers every 1000 ms to 127.0.0.1:9000
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=1).
[Cashier:9001] Serving customer...

--- Supermarket Status ---
Queue 1: 1 customers (served: 0)
Queue 2: 0 customers (served: 0)
-----
[Generator] New customer sent.
[Server] Customer assigned to cashier 2 (queue=1).
[Cashier:9002] Serving customer...
[Cashier:9001] Done.
[Server] Cashier 1 done. Remaining queue=0.
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=1).
[Cashier:9001] Serving customer...

--- Supermarket Status ---
Queue 1: 1 customers (served: 1)
Queue 2: 1 customers (served: 0)
-----
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=2).
[Cashier:9002] Done.
[Server] Cashier 2 done. Remaining queue=0.
[Cashier:9001] Done.
[Server] Cashier 1 done. Remaining queue=1.
[Cashier:9001] Serving customer...
[Generator] New customer sent.
[Server] Customer assigned to cashier 2 (queue=1).
[Cashier:9002] Serving customer...

--- Supermarket Status ---
Queue 1: 1 customers (served: 2)
Queue 2: 1 customers (served: 1)
-----
^C[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=2).
Процессы остановлены
C:\Users\user> netstat -nbt | findstr 127.0.0.1:9000
```

Видим, как появилось клиентское отображение статуса супермаркета.

## Требования на выполнение задачи на 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.

Доработаем программы сервера и клиента.

Для каждого нового клиента сервер создает поток, который обрабатывает запросы от этого клиента - `pthread_create` используется для создания нового потока, который будет обрабатывать запросы.

Когда клиент подключается, сервер создает поток для него, и этот поток будет работать с клиентом независимо от других клиентов, а когда отключается, поток завершает работу

Сервер отправляет текущее состояние очередей и обслуженных клиентов клиентам по запросу (например, при получении команды "STATUS"). Для каждого клиента создается структура `ClientData`, которая хранит сокет, адрес клиента и размер его адреса - эта структура передается в поток для обработки запросов

Как итог, всё точно так же корректно работает:

```
1: 0  (: 1)
2: 1  (: 0)
-----
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=1).
[Cashier:9001] Serving customer...
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=2).
[Cashier:9002] Done.
[Server] Cashier 2 done. Remaining queue=0.
[Cashier:9001] Done.
[Server] Cashier 1 done. Remaining queue=1.
[Cashier:9001] Serving customer...
[Generator] New customer sent.
[Server] Customer assigned to cashier 2 (queue=1).
[Cashier:9002] Serving customer...

---
1: 1  (: 2)
2: 1  (: 1)
-----
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=2).
[Cashier:9001] Done.
[Cashier:9001] Serving customer...
[Server] Cashier 1 done. Remaining queue=1.
[Generator] New customer sent.
[Server] Customer assigned to cashier 1 (queue=2).

---
1: 2  (: 3)
2: 1  (: 1)
-----
[Cashier:9002] Done.
[Server] Cashier 2 done. Remaining queue=0.
[Generator] New customer sent.
[Server] Customer assigned to cashier 2 (queue=1).
[Cashier:9002] Serving customer...
^СПроцессы остановлены
diana@LAPTOP-NJFMCILS:/mnt/d/Документы Важные ДЗ/ОСИ/ОСИ_ИДЗ/ОСИ_Ид
```

При этом имеет возможность поддерживать нескольких клиентов.

## Требования на выполнение задачи на 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**

## Что теперь:

Теперь мы можем подключать несколько клиентов прямо в консоли.

Сервер продолжает работать.

[illegible]

Я не убирала вывод генератора, чтобы было понятно, что все процессы продолжают происходить.

Теперь после команды `старт` запускается новый клиент для отображения информации.

```
ctrlC.  
[Generator] New customer sent.  
[Generator] New customer sent.  
start  
Запуск клиента для отображения информации...  
Введите команду (start - запуск клиента, stop - остановка клиента).  
[Generator] New customer sent.  
[Generator] New customer sent.  
[Generator] New customer sent.  
[Generator] New customer sent.  
[Generator] New customer sent.  
stop  
Остановлен клиент с PID 1732  
Введите команду (start - запуск клиента, stop - остановка клиента).  
[Generator] New customer sent.  
[Generator] New customer sent.  
[Generator] New customer sent.
```

Мы можем запустить несколько клиентов, а по команде stop клиенты запущенные последними будут останавливаться.

По команде quit (после нее необходимо выйти через ctrlC) завершаются все процессы.

Таким образом, мы можем добавлять и удалять клиенты в режиме реального времени!! Доработать потом сами клиенты для вывода подробной информации не составит никакого труда.

**Код лежит в соответствующей папке**

## Требования на выполнение задачи 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**

Для этого:

- Будем отслеживать все подключенные клиенты в массиве
- При завершении работы сервера будем отправлять клиентам команду для отключения
- Сервер будет отслеживать завершение всех клиентских потоков и корректно завершать их

```

2 while (1) {
3     ClientData *data = (ClientData *)malloc(sizeof(ClientData));
4     data->sock = sock;
5     data->addrlen = sizeof(data->client_addr);
6     ssize_t n = recvfrom(sock, NULL, 0, 0,
7                         (struct sockaddr *)&data->client_addr, &data->addrlen);
8     if (n < 0) {
9         perror("recvfrom");
10        continue;
11    }
12
13    pthread_mutex_lock(&client_lock);
14    clients[client_count++] = data;
15    pthread_mutex_unlock(&client_lock);
16
17    pthread_create(&data->thread_id, NULL, handle_client, data);
18    pthread_detach(
19        data->thread_id);
20 }
21 stop_all_clients();

```

Добавили функцию остановки всех клиентов

```

recvfrom(sock, buf, BUF_SIZE, 0, (struct sockaddr *)&server_addr, &addr_len);
if (strcmp(buf, "DISCONNECT") == 0) {
    printf("Server has disconnected. Closing client.\n");
}

```

А в клиенте корректно обрабатываем команду дисконнект.

До этого у нас было миллион процессов:

```

diana@LAPTOP-NJFMCILS:/mnt/d/Документы Важные ДЗ/ОСИ/ОСИ_ИДЗ/ОСИ_Идз4/dz-9$ ps aux | grep -E 'server|cashier|generator|client'
root      7  0.0  0.0  2476  196 ?        S1   00:01   0:00 plan9 --control-socket 6 --log-level 4 --server-fd 7 --pipe-fd 9 --log-truncate
diana    1498  0.0  0.0  2552  1100 pts/0    S   01:35   0:00 ./client 127.0.0.1 9000
diana    1592  0.0  0.0  2552  1004 pts/0    S   01:38   0:00 ./client 127.0.0.1 9000
diana    1643  0.0  0.0  2552  1100 pts/0    S   01:40   0:00 ./client 127.0.0.1 9000
diana    1646  0.0  0.0  2552  1032 pts/0    S   01:40   0:00 ./client 127.0.0.1 9000
diana    1647  0.0  0.0  2552  1088 pts/0    S   01:40   0:00 ./client 127.0.0.1 9000
diana    1650  0.0  0.0  2552  1008 pts/0    S   01:40   0:00 ./client 127.0.0.1 9000
diana    1678  0.0  0.0  2552  1052 pts/0    S   01:41   0:00 ./client 127.0.0.1 9000
diana    1900  0.0  0.0  2552  1100 pts/0    S   01:54   0:00 ./client 127.0.0.1 9000
diana    1905  0.0  0.0  2552  1040 pts/0    S   01:55   0:00 ./client 127.0.0.1 9000
diana    1906  0.0  0.0  2552  1064 pts/0    S   01:55   0:00 ./client 127.0.0.1 9000
diana    1924  0.0  0.0  4104  1912 pts/0    S+  01:56   0:00 grep --color=auto -E server|cashier|generator|client

```

Теперь же процессы клиентов убиваются вместе с сервером:

```

diana@LAPTOP-NJFMCILS:/mnt/d/Документы Важные ДЗ/ОСИ/ОСИ_ИДЗ/ОСИ_Идз4/dz-9$ ps aux | grep -E 'server|cashier|generator|client'
root      7  0.0  0.0  2476  196 ?        S1   00:01   0:00 plan9 --control-socket 6 --log-level 4 --server-fd 7 --pipe-fd 9 --log-truncate
diana    1929  0.0  0.0  4104  2072 pts/0    S+  01:56   0:00 grep --color=auto -E server|cashier|generator|client
diana@LAPTOP-NJFMCILS:/mnt/d/Документы Важные ДЗ/ОСИ/ОСИ_ИДЗ/ОСИ_Идз4/dz-9$

```

**ГООООЛ! Победа!**

**Спасибо за уделённое проверке время.**

