Национальный исследовательский университет “Высшая школа экономики”.

Факультет компьютерных наук. Программная инженерия.

Операционные системы.

Индивидуальное домашнее задание №4 студента группы БПИ213 Абрамова Александра Сергеевича.

Вариант 14.

14. **Задача о гостинице – 3 (дамы и джентльмены).** В гостинице 10 номеров рассчитаны на одного человека и 15 номеров рассчитаны на двух человек. В гостиницу случайно приходят клиенты–дамы и клиенты–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 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.*

СОДЕРЖАНИЕ

СЦЕНАРИЙ РЕШАЕМОЙ ЗАДАЧИ

В условии задачи представлено две сущности: отель и посетитель, между которыми происходит взаимодействие следующим образом:

1. Отель и посетители представляются в виде независимых программ, взаимодействующих с использованием протокола UDP.
2. При запуске отеля происходит инициализация комнат и UDP-сервера. После этого программа с помощью системного вызова recvfrom ожидает запросов посетителей, обработка которых производится последовательно без создания дополнительных процессов.
3. При запуске посетителя происходит создание UDP-сокета для взаимодействия с отелем (сервером) путём отправки запросов и получения ответов с помощью соответствующих системных вызовов.
4. Для получения комнаты посетитель должен в первом сообщении отправить серверу свой пол и время, в течение которого он планирует занимать комнату. При этом сервер ищет подходящую комнату, записывает в нее посетителя и отправляет клиенту успешный статус с идентификатором запроса и номером комнаты. В случае отсутствия подходящих комнат клиент получает сообщение с нулевой комнатой.
5. При получении ответа пользователь либо уходит (если комната не найдена), либо заселяется и “спит” указанное время. При этом отель может в любой момент закрыться и выгнать всех посетителей: для этого сервер отправляет каждому посетителю сообщение о необходимости немедленного завершения.
6. При успешном окончании сна посетитель сообщает отелю о выселении отправкой соответствующего запроса со своим номером и номером занимаемой комнаты. При этом сервер обрабатывает запрос и подготавливает комнату к приходу нового посетителя.

Во избежание нарушения работы отеля в связи с некорректными действиями посетителей, отель дополнительно следит за временем проживания клиентов: если время, отведённое клиенту, истекло, а запроса на выселение не поступило, то отель его выгоняет, сообщая об этом клиенту отправкой соответствующего сообщения.

Дополнительные комментарии по реализации описанного протокола представлены в исходном коде программ.

СОДЕРЖАНИЕ РАБОТЫ

В результате выполнения работы была получена одна программа, удовлетворяющая всем поставленным требованиям одновременно:

1. Исходный код программы представлен в директории `src`, описание структуры которого приведено в соответствующем разделе настоящего отчёта.
2. Исполняемые файлы программы при сборке размещаются в директории `bin`.
3. Выходные данные программы на составленных тестах представлены в директории `output`
4. Тестирование программы производилось с помощью скрипта `run.mjs`, размещённого в директории `testing`. При исполнении скрипт производит запуск программы на входных данных, указываемых вторым аргументом командной строки, каждые 250 миллисекунд создаёт наблюдателей за процессом, а также завершает работу путём отправки процессу отеля сигнала завершения $SIGINT$ по окончании работы всех посетителей. При этом производится запись выходных данных всех запущенных процессов в соответствующие текстовые файлы директории `output`. Файлы `small_test.txt` и `big_test.txt` директории `testing` содержат тестовые наборы данных, описывающие посетителей в следующем формате: первая строка указывает пол посетителя (m для мужчин и f для женщин), а вторая строка - время его пребывания в отеле в секундах (от 1 до 15).
5. `Makefile` описывает основные алгоритмы сборки и тестового запуска программ. Цели $compile\_visitor$, $compile\_hotel$ и $compile\_logger$ собирают соответствующие подпрограммы, а цели $visitor$, $hotel$ и $logger$ запускают их в режиме локального тестирования с использованием портов 8000 для запуска отеля и 9000 для организации многоадресной рассылки. Цель $compile\_all$ собирает сразу все упомянутые подпрограммы. Цели $test\_small$ и $test\_big$ запускают тестирование программы на соответствующих тестах, а цель $test$ последовательно запускает программу на обоих наборах тестовых данных. При компиляции программы используется утилита `gcc` со следующими параметрами:
   5.1. `-O2` позволяет компилятору производить оптимизации программы для более эффективной её работы
   5.2. `-fsanitize=address,undefined` указывает компилятору на необходимость добавления в код программы дополнительных инструкций, позволяющих отлавливать ошибки во время её выполнения, которые могут приводить к некорректному поведению

5.3. `-fno-sanitize-recover=all` требует немедленного завершения программы с ненулевым кодом возврата в случае обнаружения любой ошибки во время выполнения

5.4. `-std=gnu17` указывает версию спецификации языка C, которую необходимо использовать при компиляции - `C17`

5.5. `-Werror` для прекращения процесса сборки программы с ошибкой в случае обнаружения любого предупреждения компилятора

5.6. `-Wall` и `-Wextra` для включения всех предупреждений компилятора.

ПРОГРАММА

Исходный код программы представлен в директории `src`:

1. Файл `protocol.h` определяет основные значения, структуры и функции, необходимые как серверу, так и клиентам. Их объявления представлены в файле `protocol.c`
2. В поддиректории `hotel` приведена реализация программы-отеля. В качестве входных данных в виде соответствующих аргументов командной строки программе должны быть переданы порт UDP-сервера, а также адрес и порт для многоадресной рассылки информации о работе программы.
   2.1. Входная точка подпрограммы определена в файле `index.c`, где происходит инициализация всех необходимых элементов и обработка запросов.
   2.2. В директории `rooms` реализован ряд функций для удобства управления хранилищем комнат отеля: его создания и удаления, а также регистрации и ухода пользователей.
   2.3. В директории `log` реализован ряд методов, позволяющих удобно производить вывод информации и её многоадресную рассылку по протоколу UDP
3. В поддиректории `visitor` приведена реализация программы-посетителя, которая состоит из одного файла с исходным кодом - `index.c`. В качестве входных данных программе должны быть переданы IP-адрес сервера, порт сервера, пол посетителя (m или f), а также время пребывания посетителя в отеле в секундах с помощью соответствующих аргументов командной строки.
4. Поддиректория `logger` содержит исходный код одноимённой подпрограммы, которая отображает поведение моделируемой системы путём вывода в стандартный поток всех получаемых от сервера по многоадресной рассылке сообщений. В качестве входных данных программе должны быть переданы IP-адрес и порт многоадресной рассылки в виде первого и второго аргументов командной строки соответственно.

При запуске программы важно, чтобы процессы посетителей были запущены строго после процесса сервера. Корректное поведение программы в ином случае не гарантируется.

Подпрограммы отеля и посетителей отображают информацию, которая им поступает во время работы, поэтому программа удовлетворяет требованиям 4 - 5.

Подпрограмма $logger$, подключаемая к серверу, осуществляет отображение комплексной информации о выполнении приложения в целом, поэтому программа удовлетворяет требованиям 6 - 7. При этом без нарушения работы сервера возможно подключение неограниченного числа таких клиентов-наблюдателей, в том числе с независимых компьютеров. Следовательно, программа удовлетворяет требованиям 8. Более того, возможно динамическое отключение и подключение

клиентов во время работы сервера без нарушения его работоспособности. Значит, программа удовлетворяет и требованиям 9.

При завершении работы сервера, все подключенные клиенты также корректно завершаются: получение отелем сигнала прерывания приводит к отправке всем посетителям сообщения о немедленном выселении, а наблюдателям - сообщения $The\ End$. Это приводит к корректному отключению всех клиентов. Таким образом, программа удовлетворяет и требованиям 10.