Национальный исследовательский университет “Высшая школа экономики”.

Факультет компьютерных наук. Программная инженерия.

Операционные системы.

Индивидуальное домашнее задание №3 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 TCP.
2. При запуске отеля происходит инициализация комнат, TCP-сервера и необходимых примитивов синхронизации процессов. После этого программа ожидает посетителей с помощью системного вызова accept.
3. При запуске посетителя происходит создание соединения с отелем (сервером) с помощью соответствующих системных вызовов.
4. Для “общения” с каждым посетителем сервер создаёт дочерние процессы, по одному на посетителя. При этом для синхронизации доступа к информации о комнатах отеля и для вывода данных в стандартный поток используется разделяемая память, обращение к которой контролируется с помощью семафоров.
5. Для получения комнаты посетитель должен отправить серверу свой пол в качестве первого сообщения. При этом сервер ищет подходящую комнату, записывает в нее посетителя и отправляет клиенту успешный статус. В случае отсутствия подходящих комнат клиент получает статус ошибки.
6. При получении статуса ошибки посетитель завершается - уходит. Иначе посетитель заселяется и “спит” указанное время.
7. По окончании сна клиент уходит из гостиницы, что выражается завершением соединения с сервером.
8. При закрытии соединения с клиентом сервер подготавливает комнату к приходу нового посетителя и завершает дочерний процесс, созданный для обработки посетителя.

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

СОДЕРЖАНИЕ РАБОТЫ

Каталог `4 - 5` содержит программу, реализующую требования на оценки 4 - 5.

Каталог `6 - 10` содержит программу, удовлетворяющую требованиям на оценки 6 - 7, 8, 9 и 10.

Для удобства сборки и тестирования программ также представлен файл `package.json`, описывающий скрипты утилиты `npm`, которые использовались при разработке программы. В частности, реализованы следующие скрипты:

1. `compile-4-5` и `compile-6-10`, собирающие соответствующую программу.
2. `small-4-5` и `small-6-10`, производящие тестирование программы на маленьком тесте (4 посетителя) в локальной сети
3. `big-4-5` и `big-6-10`, производящие тестирование программы на большом тесте (50 посетителей) в локальной сети
4. `4-5` и `6-10`, собирающие и производящие полное тестирование программы (на обоих тестах) в локальной сети
5. `all`, производящий сборку и полное тестирование обеих программ
6. `clear`, удаляющий директории `bin` и `output` программ

ПРОГРАММА 4 - 5

Программа `4 - 5` представлена в одноимённой директории, которая содержит следующие элементы:

1. Директория `src`, содержащая исходный код программы
   1.1. Файл `protocol.h` описывает основные значения, необходимые как серверу, так и клиентам.
   1.2. В папке `visitor` приведена реализация программы-посетителя, которая состоит из одного файла с исходным кодом - `visitor.c`. В качестве входных данных программе должен быть передан 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.
   1.3. В папке `hotel` приведена реализация программы-отеля. В качестве входных данных программе должен быть передан порт сервера в виде единственного аргумента командной строки.
      1.3.1. Входная точка программы определена в файле `index.c`, где происходит инициализация всех необходимых элементов и обработка запросов.
      1.3.2. В директории `utils` представлены реализации синхропримитивов - семафоров (`sem`) и разделяемой памяти (`shm`).
      1.3.3. В директории `rooms` представлен ряд функций для удобства управления хранилищем комнат отеля: его создания и удаления, а также регистрации и ухода пользователей.
      1.3.4. В директории `log` реализован ряд методов, позволяющих удобно производить вывод информации в стандартный поток. В частности, функция `log_string` реализует вывод строки непосредственно.
2. Директория `testing`, с помощью которой производилось тестирование программ. Файлы `small_test.txt` и `big_test.txt` содержат наборы данных, на которых тестировалась программа. Каждый файл содержит описания 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). Файл `run.js` предназначен для удобства тестирования и запускает соответствующую программу на входных данных, указываемых вторым аргументом командной строки, а также завершает работу отправкой сигнала завершения $SIGINT$ серверу.
3. Директория `output`, где представлены выходные данные программы на соответствующих тестах
4. Создаваемая при сборке программы директория `bin` с исполняемыми файлами программы.

5. Файл `compile.sh`, производящий сборку программы. Получаемые исполняемые файлы сохраняются в директории `bin`. Для компиляции программы используется утилита `gcc` со следующими параметрами:
   5.1. `-O2`, позволяющий компилятору производить оптимизации программы для более эффективной работы
   5.2. `-fsanitize=address,undefined`, указывающий 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`, требующий немедленного завершения программы с ненулевым кодом возврата в случае обнаружения любой ошибки во время выполнения
   5.4. `-std=gnu17`, указывающий 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.
6. Скрипт `hotel.sh` производит запуск программыс отеля с использованием порта 8000
7. Скрипт `visitor.sh` производит запуск программы посетителя мужского пола, желающего пребывать в отеле 3 секунды, с подключением к локальному серверу на порту 8000

ПРОГРАММА 6 - 10

Для реализации возможности подключения к программе клиентов-наблюдателей были внесены следующие изменения:

1. Добавлена программа-наблюдатель `logger`, которая при запуске подключается к серверу и выводит в стандартный поток вывода все получаемые от сервера сообщения. В качестве входных данных программе должны быть переданы IP-адрес и порт сервера в виде первого и второго аргументов командной строки соответственно. Для удобства её сборки и запуска были внесены соответствующие изменения в скрипт `compile.sh`, а также создан скрипт `logger.sh` для запуска наблюдателя.
2. Для тестирования работы наблюдателей соответствующие изменения были внесены в программу `run.js`, которая во время работы помимо посетителей создаёт наблюдателей каждые 250 миллисекунд, а при завершении работы записывает в соответствующие файлы их вывод.
3. Для поддержки клиентов-наблюдателей изменения были внесены и в программу сервера:
   3.1. В директорию `utils` добавлена реализация очереди сообщений для синхронизации процессов.
   3.2. Объект `Logger` был дополнен хранилищем файловых дескрипторов клиентов-наблюдателей, а также очередью сообщений и семафором для корректной передачи сообщений клиентам-наблюдателям.
   3.3. При установлении соединения с сервером клиент должен в первую очередь сообщить свой тип: посетитель или наблюдатель. В связи с этим соответствующие изменения были внесены и в программу посетителя. При подключении наблюдателя его файловый дескриптор сохраняется в поле `destinations` объекта `Logger`, а при подключении посетителя взаимодействие происходит описанным ранее образом без значительных изменений.
   3.4. Для корректного сбора сообщений из всех дочерних процессов был реализован следующий алгоритм:
      3.4.1. При необходимости вывода информации дочерний процесс размещает сообщение в очереди сообщений объекта `Logger`, а также отправляет родительскому процессу сигнал $SIGUSR1$.
      3.4.2. При получении сигнала $SIGUSR1$ родительский процесс считывает одно сообщение из очереди, выводит его на стандартный поток, а также направляет всем подключенным наблюдателям.

3.4.3. При этом с помощью семафора обеспечивается корректная передача сообщений: в любой момент времени в очереди сообщений может находиться не более одного сообщения, причём его размещение в очередь возможно только тогда, когда родительский процесс не выполняет никаких операций, то есть ждёт очередного соединения.

Таким образом, программа удовлетворяет требованиям на 6 - 7 и 8.

Обе реализованные программы (4 - 5 и 6 - 10) удовлетворяют требованиям на 9 и 10: допускают подключение клиентов на любом этапе работы, а также обеспечивают их корректное завершение при отключении сервера. При этом завершение работы наблюдателя происходит сразу после отключения сервера, в то время как программа посетителя может продолжить работу до завершения “сна”, после чего также корректно завершится.