Лабораторная работа 03. Инструменты разработки и тестирования, СКВ.

Разработать прототип простого эмулятора файл-сервера. Проект реализуется в 2 этапа

Этап 1 – Сервер и клиент на сокетах. Обработка текстовых файлов.

Клиентская программа в интерфейсе CLI запрашивает у пользователя действие, подключается к серверу, получает ответ, печатает его в читаемом формате и завершает работу.

Клиентская программа должна:

1. Предложить пользователю ввести действие.
2. Запросить у пользователя имя файла, который будет создан, отправлен или удален.
3. Предложить пользователю ввести содержимое файла (если применимо).
4. Отправить запрос на сервер и получите ответ от сервера.
5. Распечатать соответствующее сообщение после получения ответа.
6. Отключиться от сервера и завершить работу.

На этом этапе клиентская программа должна выполнять только один рабочий цикл.

Для управления файлами использовать аналог упрощенных HTTP-запросов.
Запрос клиента должен начинаться с `GET`, `PUT` или `DELETE`. Ответ сервера, как и в HTTP, должен начинаться с кода состояния операции. Например, код `200` означает, что операция прошла успешно, и `404` означает, что ресурс не найден.

Для создания нового файла на сервере клиент отправляет запрос `PUT NAME DATA`,
где `NAME` указывается имя файла без пробелов и табуляции и `DATA` указываются текстовые данные.
Если файл с таким именем уже существует, сервер должен ответить кодом `403`.
Если файл на сервере создан успешно, сервер отвечает кодом `200`.
В этом случае клиент у себя должен распечатать `The response says that the file was created!`.
В противном случае он должен напечатать `The response says that creating the file was forbidden!`

Чтобы получить файл с сервера, клиент должен отправить запрос `GET NAME`, где `NAME` имя файла.
Если файла с таким именем не существует, сервер должен ответить кодом `404`.
Если файл с таким именем существует, сервер отвечает кодом `200`, одним пробелом и содержимым файла.
В этом случае клиент должен напечатать `The content of the file is: FILE_CONTENT`,
где `FILE_CONTENT`— содержимое запрошенного файла.
В противном случае он должен напечатать `The response says that the file was not found!`

Чтобы удалить файл с сервера, клиент должен отправить запрос `DELETE NAME`,
где `NAME` имя файла.
Если файла с таким именем не существует, сервер должен ответить кодом `404`.
Если файл успешно удален, сервер отвечает кодом `200`.
В этом случае клиент должен распечатать `The response says that the file was successfully deleted!`.
В противном случае он должен напечатать `The response says that the file was not found!`

Серверная программа, обрабатывает запросы один за другим в одном цикле. На этом этапе предполагается что запросы обрабатываются быстро и нет необходимости в параллельном выполнении.

Сервер должен размещать файлы клиента в папке *…/server/data/* .

Для завершения работы сервера реализовать его остановку по команде клиента `exit`

Серверная программа должна:

1. Распечатать `Server started!` при запуске программы.
2. Получите запрос от клиента и ответьте соответствующим образом.

- Для `PUT` запроса отправить код состояния `200`, если файл создан успешно; в противном случае отправить код состояния `403`.
- Для `GET` запроса отправить код состояния `200` и `FILE_CONTENT` разделить его одним пробелом, если файл существует; в противном случае отправить код состояния `404`.
- Для `DELETE` запроса отправьте код состояния `200`, если файл успешно удален; в противном случае отправить код состояния `404`.

3. Серверная программа не должна завершать работу до тех пор, пока не получит команду `exit`.

Примеры

Символ «больше», за которым следует пробел (>), представляет ввод пользователя (это не часть ввода).

Первое выполнение клиентской программы должно дать следующий результат:

```
Enter action (1 - get a file, 2 - create a file, 3 - delete a file): > 2
Enter filename: > 123.txt
Enter file content: > This is the first file on the server!
The request was sent.
The response says that file was created!
```

Затем сервер должен создать на диске файл с текстом «*Это первый файл на сервере!*».
Доступ к файлу должен быть возможен даже после перезапуска сервера.

После второго выполнения вывод клиентской программы:

```
Enter action (1 - get a file, 2 - create a file, 3 - delete a file): > 1
Enter filename: > 123.txt
The request was sent.
The content of the file is: This is the first file on the server!
```

Пример вывода после удаления файла:

```
Enter action (1 - get a file, 2 - create a file, 3 - delete a file): > 3
Enter filename: > 123.txt
The request was sent.
The response says that the file was successfully deleted!
```

После повторной попытки удалить тот же файл результат должен быть следующим:

```
Enter action (1 - get a file, 2 - create a file, 3 - delete a file): > 3
Enter filename: > 123.txt
The request was sent.
The response says that the file was not found!
```

После попытки получить несуществующий файл вывод должен быть следующим:

```
Enter action (1 - get a file, 2 - create a file, 3 - delete a file): > 1
Enter filename: > file_that_doesnt_exist.txt
The request was sent.
The response says that the file was not found!
```

Пример вывода после остановки сервера:

```
Enter action (1 - get a file, 2 - create a file, 3 - delete a file): > exit
The request was sent.
```

Этап 2 – Многопоточный сервер. Обработка произвольных файлов.

Клиент отправляет на сервер настоящие текстовые файлы и файлы изображений. Сервер хранит файлы и отправляет их обратно по запросу.

Написать клиентскую программу, предлагающую пользователям действия.

1. Для действия `GET` и `DELETE` спросить пользователя, хочет ли он `GET` или `DELETE` файл `BY_ID` или `BY_NAME` (не требуется для `PUT`).
2. Предложить пользователю ввести содержимое файла (если применимо).
3. Отправьте запрос на сервер и получите ответ от сервера.
4. После получения ответа распечатать соответствующее сообщение и спросить пользователя, где он хотел бы сохранить полученный файл (если применимо).
5. Отключиться от сервера и завершить работу.

Если они хотят сохранить файл на сервере, программа должна спросить пользователя, какой файл из папки *../client/data* необходимо сохранить.
После этого пользователю следует указать имя файла (имя не должно содержать пробелов и табуляций). Если пользователь не хочет указывать имя, ему следует просто нажать, Enter ничего не вводя.
Сервер должен сгенерировать уникальное имя для этого файла и отправить обратно идентификатор.
На сервере файл должен быть сохранен в папке *.../server/data/* .

Сервер должен иметь возможность распознавать каждый файл по его уникальному идентификатору. Если файл создан успешно, сервер должен вывести после кода целочисленный идентификатор 200 и один пробел. Если создание файла не удалось, идентификатор не требуется.

Клиент может получить доступ к файлу на сервере, используя его идентификатор или имя файла.
Для этого после команды GET или DELETE указать, надо ли использовать идентификатор файла или имя. Использовать BY_ID и BY_NAME в качестве ключевых слов.
Каждый раз, когда надо получить файл с сервера, вы клиент может написать GET BY_ID 12 либо GET BY_NAME filename.txt. То же самое относится и к DELETE BY_ID и DELETE BY_NAME.
Обратите внимание, что для метода PUT эти ключевые слова не нужны, он просто сохраняет новый файл на сервере, и сервер создает и отправляет новый идентификатор файла.

Если пользователь хочет получить файл, клиентская программа должна спросить, хочет ли пользователь использовать идентификатор или имя файла. После ввода идентификатора или имени пользователю необходимо указать имя, под которым следует сохранить файл. Файл должен быть сохранен в папке *.../client/data/* .
Если пользователь хочет удалить файл, клиентская программа должна спросить, хочет ли пользователь

использовать идентификатор или имя файла. После ввода идентификатора или имени программа должна отправить запрос на сервер.

Процесс сохранения файлов большего размера может занять некоторое время, поэтому следует использовать параллельный подход.
Обратите внимание, назначенные файлам идентификаторы сервер должен сохранять, они должны быть доступны и после перезагрузки сервера.
Учитывайте также, что сопоставление идентификатора с именем файла должно использоваться синхронно, если к нему имеют доступ разные потоки.

Серверная программа должна:

1. Распечатать `Server started!` сообщение при запуске программы.
2. Получите запрос от клиента и ответьте соответствующим образом.
3. Отправить ответ в зависимости от типа запроса:

- Для `PUT` запроса отправьте код состояния `200` и уникальный код `INTEGER IDENTIFIER`, разделенные одним пробелом, если файл создан успешно; в противном случае отправить код состояния `403`.
- Для `GET` запроса отправить код состояния `200` и `FILE_CONTENT` разделить его одним пробелом, если файл существует; в противном случае отправьте `404`код состояния.
- Для `DELETE` запроса отправить код состояния `200`, если файл успешно удален; в противном случае отправить код состояния `404`.

4. Серверная программа не должна завершать работу до тех пор, пока не получит `exit` команду.

Примеры

Символ «больше», за которым следует пробел (`> `), представляет ввод пользователя, это не часть ввода.

Пример 1

```
Enter action (1 - get a file, 2 - save a file, 3 - delete a file): > 2
Enter name of the file: > my_cat.jpg
Enter name of the file to be saved on server: >
The request was sent.
Response says that file is saved! ID = 23
```

Пример 2

```
Enter action (1 - get a file, 2 - save a file, 3 - delete a file): > 1
Do you want to get the file by name or by id (1 - name, 2 - id): > 2
Enter id: > 23
The request was sent.
The file was downloaded! Specify a name for it: > cat.jpg
File saved on the hard drive!
```

Пример 3

```
Enter action (1 - get a file, 2 - save a file, 3 - delete a file): > 3
Do you want to delete the file by name or by id (1 - name, 2 - id): > 2
Enter id: > 23
The request was sent.
The response says that this file was deleted successfully!
```

Пример 4

```
Enter action (1 - get a file, 2 - save a file, 3 - delete a file): > 3
Do you want to delete the file by name or by id (1 - name, 2 - id): > 2
Enter id: > 23
The request was sent.
The response says that this file is not found!
```

Пример 5

```
Enter action (1 - get a file, 2 - save a file, 3 - delete a file): > exit
The request was sent.
```

Документация для этапа 2:

ТЗ по ГОСТ 19.201 + UML-диаграммы для спецификации структуры и поведения приложения.