Мережеве програмування в середовищі Unix
Лабораторна робота №3
Багатопроцесний ітеративний TCP клієнт-сервер

Мережеве програмування потребує вирішення двох задач. Перша задача полягає в розробленні та реалізації протоколу рівня застосунку, використовуючи який клієнт та сервер комунікують. Друга задача полягає в розробці частин клієнта та сервера, які реалізують комунікацію. Клієнт – це програма, яка ініціює мережеве з'єднання. Сервер – це програма, яка очікує на запит на встановлення мережевого з'єднання. Клієнт та сервер зазвичай використовують наявні транспортні протоколи (наприклад, TCP або UDP) для мережевих з'єднань. Після встановлення мережевого з'єднання, поділ програм на клієнт та сервер умовний. Сервер може працювати тільки з одним клієнтом або з кількома клієнтами одночасно (одночасно не обов'язково значить те саме, що паралельно). Так само може працювати клієнт.

Інформація в протоколі рівня застосунку зазвичай відправляється повідомленнями, які складаються з заголовку (може бути кілька заголовків, але перший заголовок завжди має той самий формат) та даних змінної довжини (дані також можуть бути опціональні). Повідомлення може бути відправлено кількома сегментами транспортного протоколу, тому програма, яка отримує ці сегменти, має збирати повідомлення. Протокол рівня застосунку можна представити діаграмою, в якій є стани та дуги з умовами переходів між станами. Програмі достатньо знати поточний стан протоколу рівня застосунку щоб визначити який буде наступний стан при отриманні чергового повідомлення. Вміст заголовку або заголовків повідомлення визначають умови переходів між станами.

Ця лабораторна робота присвячена практичному використанню функцій POSIX для роботи з TCP з'єднаннями. Завдання полягає в розробленні протоколу рівня застосунку, клієнта, кількох реалізацій серверів.

Завдання на роботу

Розробити клієнт та сервер, які виконують наступне:

1. Клієнт підтримує такі аргументи командного рядка: адреса сервера, порт сервера, ім'я файлу, максимальний розмір файлу.

2. Сервер підтримує такі аргументи командного рядка: адреса сервера, порт сервера, шляхове ім'я директорії.

3. Клієнт та сервер використовують транспортний протокол TCP для мережевого з'єднання.

4. Клієнт відправляє запит серверу з ім'ям файлу, яке вказано в аргументі його командного рядка. Сервер отримує запит від клієнта, шукає звичайний файл з вказаним ім'ям у директорії, шляхове ім'я якої вказано в аргументі його командного рядка, та відправляє клієнту вміст файлу. Клієнт записує отриманий вміст у звичайний файл.

Протокол рівня застосунку має наступні характеристики:

1. Протокол має версію. Якщо версії протоколів, які використовують клієнт та сервер не збігаються, тоді з'єднання між клієнтом та сервером треба завершити.

2. Ім'я файлу в запиті клієнта повинно складатися з символів ASCII, які дозволені для імені файлу в наявній ФС (літери, цифри, знаки пунктуації і т. ін.). Максимальна довжина імені

файлу обмежена. Клієнт може надіслати які-завгодно дані замість імені файлу, сервер має перевірити коректність цих даних. Клієнт має надіслати ім’я файлу, а не шляхове ім’я.

3. Сервер відправляє клієнту інформацію чи було знайдено файл з вказаним ім’ям та його розмір, у випадку помилки серер відправляє клієнту номер помилки та завершує з’єднання. Розмір файлу не перевищує значення (2^{64} - 1) байт (тобто є 64 біт для значення розміру файлу в заголовку). У випадку помилки клієнт виводить інформацію про помилку та завершує з’єднання. Якщо розмір файлу перевищує вказаний максимальний розмір файлу в аргументі командного рядка клієнта, тоді клієнт відправляє серверу повідомлення про відмову отримувати вміст файлу, інакше клієнт відправляє серверу повідомлення про готовність отримувати вміст файлу.

4. Якщо сервер отримав повідомлення від клієнта про готовність отримувати вміст файлу, тоді він відправляє вміст файлу частинами (тобто може потребуватися кілька викликів відповідного системного виклику для відправлення вмісту файлу). Розмір частини визначається в сервері константним значенням. Відправивши весь вміст файлу, сервер завершує з’єднання. Отримавши весь вміст файлу клієнт завершує з’єднання.

Треба реалізувати наступні реалізації серверів:

1. Ітеративний сервер, який опрацьовує запити одного клієнта повністю, перед тим, як почати опрацьовувати запити наступного клієнта.

2. Паралельний сервер, який створює нові процеси для опрацювання запитів нових клієнтів. Сервер має обмеження на максимальну кількість дочірніх процесів, які опрацьовують запити клієнтів. Ця максимальна кількість вказується в аргументі командного рядка сервера. Сервер не приймає нові TCP з’єднання від клієнтів після досягнення цієї кількості.

3. Паралельний сервер, який заздалегідь створює нові процеси для опрацювання запитів клієнтів, кожний дочірній процес є ітеративним сервером, як у першому пункті. Кількість дочірніх процесів, які має створити сервер, вказується в аргументі командного рядка сервера.

Сервер не має завершувати своє виконання у випадку виникнення несистемної помилки.

Для перевірки коректності роботи програм рекомендується виводити повідомлення про дії в програмах (адреси, номери портів, вміст заголовків).

Звіт

- Опис розроблених програм (за бажанням).
- Лістинг розроблених програм.
- Приклади використання розроблених програм.

Література

Література надана у вступі до лекцій.

Лекції 1, 2, 3, 4.