Міністерство освіти і науки України
Міжнародний гуманітарний університет
Кафедра інформаційних технологій

М.А. Мірошник

ПРОГРАМУВАННЯ ІНФОКОМУНІКАЦІЙНИХ СЕРВІСІВ І СИСТЕМ

Методичні вказівки до практичних робіт
для здобувачів вищої освіти,
які навчаються за спеціальностями
галузі «Інформаційні технології»

Одеса 2025

Затверджено Вченою Радою Міжнародного гуманітарного університету.
Протокол № 5 від 20 січня 2025 р.

М.А. Мірошник Програмування інфокомунікаційних сервісів і систем. Методичні вказівки до практичних робіт для здобувачів вищої освіти, які навчаються за спеціальностями галузі «Інформаційні технології» [Електронне видання]. Кафедра інформаційних технологій Міжнародного гуманітарного університету. Одеса, 2025. 45 с.

Дисципліна «Програмування інфокомунікаційних сервісів і систем» викладається на завершальному етапі підготовки бакалаврів. Дисципліна має інтегральний характер і узагальнює знання та навички, отримані здобувачами під час вивчення дисциплін з програмування, комп'ютерних мереж, операційних систем, вебтехнологій, баз даних, безпеки програм та даних. У межах дисципліни розглядаються принципи програмної взаємодії з мережевими сервісами, методи програмної інтеграції інформаційних систем і сервісів за допомогою API та вебтехнологій, а також обмін даними між програмними компонентами з використанням форматів JSON і XML. Окрему увагу приділено взаємодії програмних застосунків з інфокомунікаційними пристроями, моніторингу роботи інфокомунікаційних систем, використанню серверних служб, консольних утиліт і скриптів для управління сервісами та питанням забезпечення безпеки програм і даних у мережевих системах.

ЗМІСТ

ТЕМА 1. ПРИНЦИПИ ПРОГРАМНОЇ ВЗАЄМОДІЇ З ІНФОКОМУНІКАЦІЙНИМИ СИСТЕМАМИ

Практична робота 1. Передавання повідомлень через мережеві сокети

Мета роботи – ознайомитися з принципами роботи мережевих сокетів, реалізувати встановлення TCP-з'єднання та передавання повідомлень між клієнтом і сервером.

Ключові положення

Мережеві сокети є базовим механізмом програмної взаємодії у розподілених системах, що дозволяє реалізувати обмін даними між процесами, які виконуються на різних вузлах мережі. Сокет визначає кінцеву точку зв'язку і ідентифікується комбінацією IP-адреси та номера порту. Використання сокетів дає змогу програмам встановлювати з'єднання, передавати дані і завершувати взаємодію у стандартизований спосіб.

У випадку TCP-з'єднання використовується модель взаємодії, орієнтована на встановлення з'єднання. Перед початком обміну даними між клієнтом і сервером виконується процедура встановлення з'єднання, яка забезпечує синхронізацію сторін і готовність до передавання інформації. Такий підхід гарантує надійність передачі, впорядкованість даних і контроль помилок.

TCP забезпечує передачу даних у вигляді потоку байтів. Це означає, що інформація не має явного поділу на повідомлення, а передається як безперервна послідовність. Визначення меж повідомлень покладається на програмний рівень. Це вимагає розробки власних механізмів структурування даних, наприклад, використання розділювачів або передачі довжини повідомлення.

Серверна частина застосунку виконує роль приймача запитів. Вона створює сокет, прив'язує його до певного порту і переходить у режим очікування підключень. Після надходження запиту від клієнта сервер приймає з'єднання і починає обмін даними. У реальних системах сервер може обслуговувати декілька клієнтів одночасно, що потребує використання багатопотокових або асинхронних механізмів.

Клієнтська частина ініціює з'єднання із сервером, використовуючи його IP-адресу і порт. Після встановлення з'єднання клієнт може надсилати дані і отримувати відповіді. Така модель є основою більшості мережевих застосунків.

Передавання даних через сокети здійснюється за допомогою операцій запису і читання. Дані передаються у вигляді масивів байтів, що забезпечує універсальність, але потребує додаткової обробки для роботи зі структурованими даними. Програміст повинен враховувати можливі затримки, часткове отримання даних і необхідність повторних операцій.

Важливим є коректне завершення з'єднання. Закриття сокета повинно виконуватися після завершення обміну даними, щоб уникнути втрати інформації або витоку ресурсів. Це є необхідною умовою стабільної роботи мережевого застосунку.

Таким чином, встановлення TCP-з'єднання і передавання повідомлень є базовими операціями у мережевому програмуванні, що забезпечують надійний обмін даними між компонентами системи.

Практичне завдання

1. Реалізувати серверний застосунок, що очікує підключення клієнтів на заданому порту.
2. Реалізувати клієнтський застосунок, що встановлює TCP-з’єднання із сервером.
3. Забезпечити передавання текстового повідомлення від клієнта до сервера.
4. Реалізувати відправлення відповіді від сервера до клієнта.
5. Вивести отримані повідомлення у консоль клієнта і сервера.
6. Дослідити поведінку системи при одночасному підключенні декількох клієнтів.
7. Реалізувати коректне закриття з’єднання після завершення обміну даними.
8. Проаналізувати процес встановлення з’єднання і передавання даних.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з принципами роботи TCP-протоколу, моделлю клієнт–сервер та поняттям сокета як кінцевої точки мережевої взаємодії. Особливу увагу слід приділити етапам встановлення TCP-з’єднання та особливостям потокової передачі даних.

На першому етапі необхідно реалізувати серверний застосунок. Для цього слід створити сокет із використанням TCP-протоколу, прив’язати його до локальної IP-адреси (або до всіх доступних інтерфейсів) і вибраного порту. Після цього сервер потрібно перевести у режим очікування підключень. У програмі має бути реалізовано приймання вхідного з’єднання та виведення повідомлення про підключення клієнта. Після встановлення з’єднання сервер повинен виконати операцію приймання даних і вивести отримане повідомлення у консоль.

На другому етапі необхідно реалізувати клієнтський застосунок. Клієнт повинен створити сокет і встановити з’єднання із сервером, використовуючи його IP-адресу і порт. Після встановлення з’єднання необхідно передати текстове повідомлення на сервер. Повідомлення повинно бути сформоване у вигляді послідовності байтів. Після цього клієнт має отримати відповідь від сервера і вивести її у консоль.

Під час реалізації обміну даними необхідно врахувати, що TCP передає інформацію у вигляді потоку. Це означає, що дані можуть надходити частинами. Тому слід реалізувати читання даних у циклі до повного отримання повідомлення або використати простий спосіб визначення меж повідомлення, наприклад фіксований розмір або спеціальний символ завершення.

На наступному етапі необхідно додати обробку помилок. Зокрема, слід передбачити ситуації, коли сервер недоступний, з’єднання переривається або виникає помилка під час передавання даних. У таких випадках програма повинна вивести повідомлення про помилку і завершити роботу коректно.

Після завершення обміну даними необхідно виконати закриття сокетів як на стороні клієнта, так і на стороні сервера. Закриття має виконуватися після завершення усіх операцій передавання і приймання даних.

Для кращого розуміння роботи системи рекомендується виконати додаткові експерименти. Зокрема, слід перевірити підключення декількох клієнтів до сервера, змінити розмір повідомлень, а також перевірити поведінку програми у випадку примусового

завершення з’єднання. Також доцільно проаналізувати, як змінюється робота системи при затримках у передаванні даних.

У процесі виконання роботи необхідно зафіксувати результати роботи програм, звернути увагу на встановлення з’єднання, передавання повідомлень і завершення взаємодії. Отримані результати слід використати для формування висновків щодо особливостей роботи TCP-з’єднання і мережевих сокетів.

Контрольні питання

1. Що таке мережевий сокет?
2. Які параметри визначають сокет?
3. У чому полягає принцип роботи TCP-з’єднання?
4. Які етапи встановлення з’єднання існують?
5. Як здійснюється передавання даних через сокети?
6. Чому TCP забезпечує надійність передачі?
7. Яку роль виконує сервер у клієнт–серверній моделі?
8. Як реалізується клієнтська частина застосунку?
9. Які проблеми можуть виникати при передаванні даних?
10. Чому важливо коректно закривати з’єднання?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис принципів роботи TCP-з’єднання;
– опис реалізації серверної частини;
– опис реалізації клієнтської частини;
– приклади передавання повідомлень;
– результати експериментів;
– висновки.

Практична робота 2. Обробка мережевих повідомлень

Мета роботи – набути навичок передавання та приймання даних через сокети, реалізувати обробку мережевих повідомлень і структурування потокових даних.

Ключові положення

У мережевих застосунках передавання даних через сокети потребує не лише встановлення з’єднання, але й організації коректної обробки повідомлень. Оскільки TCP забезпечує потокову передачу даних, програмний застосунок повинен самостійно визначати структуру повідомлень і правила їх обробки.

Мережеве повідомлення є логічною одиницею даних, що передається між компонентами системи. Воно може містити текст, числові значення або структуровані дані.

Для забезпечення коректної обробки необхідно визначити формат повідомлення і правила його інтерпретації.

Одним із підходів є використання розділювачів повідомлень. У цьому випадку кожне повідомлення завершується спеціальним символом або послідовністю символів. Інший підхід полягає у передачі довжини повідомлення перед самими даними, що дозволяє точно визначити межі.

Обробка повідомлень містить їх приймання, аналіз і формування відповіді. Програма повинна перевіряти коректність даних, обробляти помилки і забезпечувати відповідну реакцію. Це є важливим для забезпечення стабільної роботи системи.

У реальних застосунках часто використовується обробка декількох типів повідомлень. Це вимагає реалізації логіки маршрутизації, яка визначає, як саме обробляти кожне повідомлення. Такий підхід дозволяє створювати гнучкі і масштабовані системи.

Важливим є також питання буферизації даних. Оскільки дані можуть надходити частинами, необхідно накопичувати їх у буфері і обробляти лише після отримання повного повідомлення. Це дозволяє уникнути помилок і забезпечити коректність роботи.

Таким чином, обробка мережевих повідомлень є важливою складовою мережевого програмування і визначає коректність взаємодії між компонентами системи.

Практичне завдання

1. Розширити реалізацію клієнта і сервера для підтримки обміну декількома повідомленнями.
2. Реалізувати формат повідомлення із використанням розділювачів або довжини.
3. Забезпечити приймання і обробку повідомлень на стороні сервера.
4. Реалізувати формування відповіді залежно від отриманого повідомлення.
5. Додати обробку помилок при некоректних даних.
6. Реалізувати буферизацію даних для коректного приймання повідомлень.
7. Забезпечити можливість багаторазового обміну повідомленнями без розриву з’єднання.
8. Проаналізувати процес обробки повідомлень.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з особливостями потокової передачі даних у TCP та принципами організації логічних повідомлень поверх потоку байтів. Необхідно чітко усвідомити, що TCP не забезпечує розмежування повідомлень, тому визначення їх меж виконується на рівні програмного застосунку.

На першому етапі потрібно визначити формат повідомлення, який буде використовуватися у застосунку. Рекомендується обрати один із двох підходів: використання спеціального символу завершення повідомлення або передавання довжини повідомлення перед самими даними. Обраний формат необхідно зафіксувати та використовувати однаково як на стороні клієнта, так і на стороні сервера.

Далі необхідно реалізувати обробку даних на сервері. Сервер повинен приймати потік байтів, накопичувати дані у буфері та визначати момент отримання повного повідомлення відповідно до обраного формату. Після визначення меж повідомлення необхідно виконати його обробку, наприклад вивести у консоль або сформувати відповідь клієнту. Обробка

повинна бути організована у циклі для підтримки багаторазового приймання повідомлень у межах одного з'єднання.

На наступному етапі необхідно реалізувати клієнтський застосунок, який підтримує багаторазове надсилання повідомлень без розриву з'єднання. Клієнт повинен формувати повідомлення відповідно до визначеного формату, надсилати їх на сервер і отримувати відповіді. Для перевірки роботи доцільно реалізувати введення повідомлень з клавіатури.

Особливу увагу необхідно приділити буферизації даних. Оскільки повідомлення можуть надходити частинами, необхідно реалізувати механізм накопичення даних у буфері до моменту отримання повного повідомлення. При цьому слід уникати втрати або дублювання даних.

Обов'язковим елементом є обробка помилок. Необхідно передбачити ситуації отримання некоректних або неповних даних, розриву з'єднання, а також перевищення допустимого обсягу повідомлення. У таких випадках програма повинна виводити повідомлення про помилку та продовжувати роботу або коректно завершувати з'єднання.

Після реалізації необхідно провести тестування роботи застосунку. Рекомендується перевірити передавання декількох повідомлень підряд, передачу довгих повідомлень, а також роботу при частковому отриманні даних. Доцільно також перевірити поведінку системи при некоректному форматі повідомлення.

У завершенні роботи необхідно проаналізувати отримані результати, звернути увагу на правильність визначення меж повідомлень, стабільність обробки даних та стійкість програми до помилок. На основі цього слід сформулювати висновки щодо ефективності обраного підходу до організації мережевих повідомлень.

Контрольні питання

1. Що таке мережеве повідомлення?
2. Чому TCP не визначає межі повідомлень?
3. Які методи визначення меж повідомлень існують?
4. Що таке буферизація даних?
5. Як реалізується обробка повідомлень у сервері?
6. Які проблеми можуть виникати при прийманні даних?
7. Як забезпечити обробку декількох повідомлень?
8. Яку роль відіграє формат повідомлення?
9. Як реалізується обробка помилок?
10. Чому важливо тестувати мережеві застосунки?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис формату повідомлень;
– опис реалізації клієнта і сервера;
– приклади обміну повідомленнями;
– результати тестування;
– висновки.

ТЕМА 2. МЕТОДИ ПРОГРАМНОЇ ІНТЕГРАЦІЇ ЗАСОБІВ ІНФОКОМУНІКАЦІЙ

Практична робота 3. Формування HTTP-запитів

Мета роботи – дослідити принципи формування HTTP-запитів, отримання та аналізу відповідей від вебсервісів.

Ключові положення

У сучасних інфокомунікаційних системах взаємодія між програмними компонентами часто реалізується через вебсервіси, що функціонують у мережевому середовищі та забезпечують обмін даними між різними застосунками. Такий підхід дозволяє розподіляти функціональність системи між окремими модулями, що взаємодіють через стандартизовані інтерфейси. Основним протоколом, який використовується для такої взаємодії, є HTTP. Він забезпечує передачу запитів від клієнта до сервера і отримання відповідей у стандартизованому форматі. Програмний застосунок у цьому випадку виступає у ролі клієнта, який формує запити, надсилає їх до сервера і обробляє отримані результати відповідно до логіки роботи системи.

HTTP є текстовим протоколом прикладного рівня, що працює поверх транспортного протоколу TCP. Він визначає правила обміну повідомленнями між клієнтом і сервером, а також структуру цих повідомлень. Взаємодія відбувається за принципом «запит–відповідь», де клієнт ініціює запит, а сервер обробляє його і повертає результат. Така модель є основою більшості сучасних вебзастосунків і сервісів.

HTTP-запит має чітко визначену структуру, яка містить рядок запиту, заголовки і, за необхідності, тіло повідомлення. Рядок запиту містить метод, адресу ресурсу і версію протоколу. Найбільш поширеними методами є GET, POST, PUT і DELETE. Метод GET використовується для отримання даних і не змінює стан ресурсу. Метод POST застосовується для передавання даних на сервер і створення нових ресурсів. Метод PUT використовується для оновлення існуючих ресурсів, а DELETE – для їх видалення. Вибір методу визначає характер взаємодії між клієнтом і сервером і має відповідати логіці роботи сервісу.

Заголовки HTTP-запиту містять додаткову інформацію, необхідну для обробки запиту. Вони можуть містити тип даних, що передаються, параметри автентифікації, інформацію про клієнта або інші службові відомості. Тіло запиту використовується для передавання даних, наприклад у випадку використання методу POST. Дані можуть передаватися у різних форматах, зокрема у вигляді JSON або XML.

HTTP-відповідь, яку повертає сервер, також має визначену структуру. Вона містить рядок статусу, заголовки і тіло відповіді. Код статусу є важливим елементом відповіді, оскільки дозволяє визначити результат виконання запиту. Наприклад, код 200 означає успішне виконання, 404 – відсутність ресурсу, 500 – помилку на стороні сервера. Аналіз кодів статусу є необхідним для коректної обробки результатів і прийняття рішень у програмі.

Тіло відповіді містить дані, що повертаються сервером. Це можуть бути як прості текстові повідомлення, так і складні структуровані дані. Обробка цих даних є важливою частиною програмної взаємодії, оскільки саме на їх основі виконується подальша логіка

застосунку. У більшості випадків дані передаються у форматі JSON, який є зручним для обробки і широко підтримується сучасними мовами програмування.

У програмних застосунках формування HTTP-запитів зазвичай виконується за допомогою спеціалізованих бібліотек або вбудованих засобів мови програмування. Це дозволяє абстрагуватися від низькорівневих деталей роботи з мережею і зосередитися на логіці обміну даними. Такі бібліотеки забезпечують зручні інтерфейси для створення запитів, встановлення заголовків, передавання даних і обробки відповідей.

Важливим питанням є обробка помилок під час виконання HTTP-запитів. У процесі взаємодії можуть виникати різні проблеми, такі як відсутність з'єднання, перевищення часу очікування, некоректні відповіді або помилки сервера. Програмний застосунок повинен враховувати ці ситуації і коректно на них реагувати. Для цього необхідно перевіряти коди статусу, обробляти виключення і забезпечувати повторні спроби виконання запитів за необхідності.

Окрему увагу слід приділити питанню продуктивності. Часте виконання HTTP-запитів може створювати додаткове навантаження на систему і мережу. Тому у деяких випадках доцільно використовувати механізми кешування або обмеження частоти запитів. Це дозволяє підвищити ефективність роботи застосунку і зменшити затримки.

Також важливим є забезпечення безпеки під час взаємодії з вебсервісами. Для цього використовується протокол HTTPS, який забезпечує шифрування даних і захист від перехоплення. При роботі з вебсервісами необхідно перевіряти сертифікати і використовувати захищені з'єднання, особливо у випадку передавання конфіденційної інформації.

У сучасних системах HTTP-взаємодія часто використовується у межах REST-підходу, який передбачає використання стандартних методів і чіткої структури ресурсів. Це спрощує інтеграцію між системами і забезпечує уніфікований підхід до обміну даними.

Таким чином, формування HTTP-запитів і обробка відповідей є базовою задачею програмної інтеграції, що дозволяє взаємодіяти з вебсервісами. Розуміння структури запитів і відповідей, методів HTTP, принципів обробки даних і помилок є необхідним для створення ефективних і надійних програмних систем.

Практичне завдання

1. Реалізувати програму для надсилання HTTP-запиту методом GET.
2. Отримати і вивести відповідь від вебсервісу.
3. Реалізувати надсилання запиту методом POST із передачею даних.
4. Проаналізувати коди статусу відповіді.
5. Вивести заголовки відповіді.
6. Реалізувати обробку помилок при виконанні запиту.
7. Дослідити час відповіді сервера.
8. Сформулювати висновки щодо роботи HTTP-запитів.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися зі структурою HTTP-запиту і HTTP-відповіді, а також з основними методами протоколу HTTP. Необхідно чітко розуміти, з яких елементів складається запит (метод, адреса ресурсу, заголовки, тіло) і які

складові має відповідь (код статусу, заголовки, тіло). Особливу увагу слід приділити призначенню методів GET і POST, а також ролі заголовків у передаванні додаткової інформації.

На першому етапі необхідно реалізувати виконання HTTP-запиту методом GET до відкритого вебсервісу. Для цього слід обрати доступний ресурс, який повертає дані у текстовому або структурованому форматі. У програмі потрібно сформувати запит, надіслати його на сервер і отримати відповідь. Отримані дані необхідно вивести у консоль у зручному для аналізу вигляді. Додатково слід зафіксувати код статусу відповіді та переконатися, що запит виконано успішно.

На наступному етапі необхідно реалізувати HTTP-запит методом POST. Для цього потрібно сформувати тіло запиту, яке містить дані, що передаються на сервер. Дані можуть бути представлені у вигляді тексту або структурованого формату. Необхідно також правильно сформувати заголовки запиту, зокрема вказати тип переданих даних. Після надсилання запиту потрібно отримати відповідь від сервера і вивести її у консоль.

Під час виконання обох типів запитів необхідно забезпечити аналіз отриманих відповідей. Особливу увагу слід приділити кодам статусу HTTP. У програмі має бути реалізована перевірка кодів статусу і відповідна реакція на них. Наприклад, при успішному виконанні запиту слід вивести повідомлення про успіх, а у випадку помилки – повідомлення про причину збою.

Обов’язковим елементом є реалізація обробки помилок. Необхідно передбачити ситуації відсутності з’єднання, некоректної адреси ресурсу, перевищення часу очікування або отримання помилкової відповіді від сервера. У таких випадках програма повинна коректно обробляти помилки і не завершувати роботу аварійно.

Для поглибленого аналізу доцільно додатково вивести заголовки відповіді сервера. Це дозволяє отримати інформацію про тип даних, довжину відповіді та інші параметри, що характеризують роботу сервісу.

Після реалізації необхідно провести тестування програми. Рекомендується виконати декілька запитів до різних ресурсів, перевірити роботу з коректними і некоректними адресами, а також дослідити поведінку програми при різних кодах статусу. Результати виконання необхідно зафіксувати.

У завершенні роботи необхідно проаналізувати отримані результати, звернути увагу на відмінності між запитами GET і POST, особливості формування запитів і обробки відповідей. На основі цього слід сформулювати висновки щодо принципів роботи HTTP та особливостей програмної взаємодії з вебсервісами.

Контрольні питання

1. Що таке HTTP-запит?
2. Які методи HTTP існують?
3. Що містить HTTP-відповідь?
4. Яке призначення кодів статусу?
5. Як формується GET-запит?
6. У чому особливість POST-запиту?
7. Яку роль відіграють заголовки?
8. Як обробляються помилки?
9. Що таке тіло запиту?
10. Як вимірюється час відповіді?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис HTTP-запитів;
– приклади реалізації;
– результати виконання;
– висновки.

Практична робота 4. Обробка даних у форматах JSON та XML

Мета роботи – ознайомитися з форматами JSON та XML, набути навичок обробки структурованих даних у програмних застосунках.

Ключові положення

У процесі взаємодії з вебсервісами дані передаються у структурованому вигляді, що дозволяє різним програмним компонентам коректно інтерпретувати отриману інформацію незалежно від мови програмування або платформи. Структуровані формати забезпечують стандартизований спосіб подання даних, що є критично важливим для інтеграції розподілених систем. Найбільш поширеними форматами передавання даних є JSON і XML, які дозволяють представляти складні структури у текстовому вигляді та легко передавати їх через мережу.

JSON (JavaScript Object Notation) є легким текстовим форматом обміну даними, що базується на представленні інформації у вигляді пар ключ–значення. Його структура є простою і зрозумілою, що значно спрощує процес створення і обробки даних. JSON підтримує основні типи даних, такі як рядки, числа, логічні значення, масиви та вкладені об’єкти. Завдяки своїй компактності він займає менше місця при передаванні і швидше обробляється, що робить його оптимальним вибором для більшості сучасних вебзастосунків і API.

JSON широко використовується у REST-сервісах, де клієнт отримує або надсилає дані у цьому форматі. Його популярність зумовлена також тим, що він природно підтримується мовами програмування, що спрощує перетворення даних у внутрішні структури програми. Наприклад, об’єкти JSON легко відображаються у вигляді словників або асоціативних масивів, що дозволяє безпосередньо працювати з даними після їх отримання.

XML (eXtensible Markup Language) є більш формалізованим форматом, що використовує тегову структуру для опису даних. Він дозволяє створювати складні ієрархічні структури та чітко визначати зв’язки між елементами. XML є самодокументованим, оскільки теги описують зміст даних, що полегшує їх розуміння і аналіз.

Однією з важливих переваг XML є можливість валідації структури за допомогою схем. Це дозволяє перевіряти відповідність даних визначеним правилам і забезпечує їх коректність. XML широко використовується у системах, де важлива строгість структури і контроль правильності даних, наприклад у фінансових або корпоративних системах.

Разом із тим XML є більш об’ємним порівняно з JSON, оскільки містить додаткові службові елементи у вигляді тегів. Це може призводити до збільшення обсягу переданих

даних і зниження продуктивності. Тому у сучасних вебзастосунках частіше використовується JSON, тоді як XML застосовується у специфічних випадках, де потрібна формальна структура і розширені можливості опису.

Обробка структурованих даних містить декілька етапів. Першим є отримання даних у вигляді текстового повідомлення від вебсервісу. Далі виконується парсинг, тобто перетворення текстового представлення у внутрішню структуру даних, з якою може працювати програма. Для JSON це може бути перетворення у словник або об’єкт, для XML – у дерево елементів.

Після парсингу виконується доступ до окремих елементів даних. Програма повинна мати змогу отримувати значення за ключами або тегами, аналізувати структуру і виконувати необхідні операції. Це може містити відбір певних елементів, обчислення значень або перетворення даних у інший формат.

Важливим етапом є обробка помилок. Дані, отримані від вебсервісу, можуть бути некоректними або неповними. Тому необхідно перевіряти правильність структури, наявність необхідних елементів і відповідність типів даних. У випадку виявлення помилки програма повинна коректно обробити ситуацію і не завершувати роботу аварійно.

У процесі роботи з JSON і XML важливо враховувати вкладеність структур. Дані можуть містити декілька рівнів вкладеності, що ускладнює доступ до окремих елементів. Для роботи з такими структурами використовуються спеціальні методи або бібліотеки, які дозволяють ефективно здійснювати навігацію по даних.

Вибір формату залежить від вимог системи. Якщо пріоритетом є швидкість обробки і мінімальний обсяг даних, доцільно використовувати JSON. Якщо ж необхідна строгість структури, підтримка схем і розширені можливості опису, доцільніше застосовувати XML. У багатьох випадках системи підтримують обидва формати, що забезпечує гнучкість взаємодії.

У сучасних інфокомунікаційних системах обробка структурованих даних є невід’ємною частиною програмної інтеграції. Вона дозволяє об’єднувати різні сервіси, обмінюватися інформацією і забезпечувати взаємодію між компонентами системи. Розуміння принципів роботи з JSON і XML є необхідним для розробки сучасних мережевих застосунків.

Таким чином, JSON і XML є основними форматами передавання структурованих даних у мережевих системах. Вони мають різні характеристики і сфери застосування, але обидва забезпечують можливість ефективного обміну інформацією. Обробка таких даних містить парсинг, аналіз і використання інформації у програмі, що є ключовим елементом інтеграції програмних систем.

Практичне завдання

1. Отримати дані у форматі JSON з вебсервісу.
2. Виконати парсинг JSON-даних.
3. Вивести окремі елементи структури.
4. Отримати дані у форматі XML.
5. Виконати парсинг XML-документа.
6. Проаналізувати структуру XML.
7. Порівняти JSON і XML.
8. Сформулювати висновки.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з основними принципами представлення структурованих даних у форматах JSON та XML, а також зі способами їх обробки у вибраній мові програмування. Необхідно розуміти відмінності у структурі цих форматів, способах доступу до даних і особливостях їх парсингу.

На першому етапі необхідно обрати вебсервіс, що повертає дані у форматі JSON. Доцільно використати відкритий API, який надає доступ до структурованих даних. Після цього потрібно реалізувати отримання відповіді від сервісу за допомогою HTTP-запиту. Отримані дані необхідно вивести у консоль у початковому вигляді для ознайомлення зі структурою.

Далі необхідно виконати парсинг JSON-даних. Для цього слід використати відповідні засоби мови програмування або бібліотеки, які дозволяють перетворити текстове представлення у внутрішню структуру даних. Після парсингу потрібно отримати доступ до окремих елементів, наприклад до значень за ключами, і вивести їх у зручному вигляді. Особливу увагу слід приділити роботі з вкладеними структурами та масивами.

На наступному етапі необхідно виконати аналогічні дії для формату XML. Потрібно отримати XML-документ, який може бути отриманий з вебсервісу або підготовлений окремо. Далі слід виконати парсинг XML-документа та представити його у вигляді ієрархічної структури. Необхідно реалізувати доступ до елементів за тегами та вивести їх значення у консоль.

Особливу увагу під час роботи з XML слід приділити структурі документа. Необхідно проаналізувати вкладеність елементів, наявність атрибутів та ієрархію даних. Це дозволяє зрозуміти особливості представлення інформації у цьому форматі та відмінності від JSON.

Після реалізації обробки обох форматів необхідно виконати їх порівняльний аналіз. Потрібно оцінити зручність роботи з JSON і XML, складність парсингу, обсяг даних, а також швидкість обробки. На основі цього слід визначити переваги і недоліки кожного формату у контексті програмної інтеграції.

Під час виконання роботи необхідно також враховувати можливі помилки, зокрема некоректний формат даних або відсутність окремих елементів. Програма повинна коректно обробляти такі ситуації і не завершувати роботу аварійно.

У завершенні необхідно проаналізувати отримані результати, звернути увагу на відмінності у структурі і способах обробки даних, а також сформулювати висновки щодо доцільності використання кожного формату у різних типах програмних систем.

Контрольні питання

1. Що таке JSON?
2. Які основні елементи JSON?
3. Що таке XML?
4. У чому відмінність JSON і XML?
5. Що таке парсинг?
6. Як обробляються структуровані дані?
7. Які переваги JSON?
8. Які переваги XML?
9. Де використовуються ці формати?
10. Які проблеми можуть виникати при обробці?

Зміст звіту

– назва роботи;
– мета роботи;
– опис використаних форматів;
– приклади обробки;
– результати;
– висновки.

Практична робота 5. Програмна взаємодія з пристроями через інтерфейс RS-485

Мета роботи – дослідити принципи програмної взаємодії з пристроями через інтерфейс RS-485 та реалізувати обмін даними через послідовний порт.

Ключові положення

Інтерфейс RS-485 є одним із найбільш поширених стандартів фізичного рівня для організації обміну даними між пристроями у промислових та інфокомунікаційних системах. Він широко використовується у системах автоматизації, промислових контролерах, системах збору даних та мережах Інтернету речей. Основною перевагою RS-485 є його здатність забезпечувати надійний зв'язок на значних відстанях і в умовах підвищених електромагнітних завад, що робить його придатним для використання у складних середовищах.

RS-485 базується на диференціальному способі передавання сигналу, при якому інформація передається не відносно загального проводу, а як різниця потенціалів між двома лініями. Такий підхід дозволяє значно зменшити вплив зовнішніх завад і підвищити стійкість передачі даних. Саме ця особливість робить RS-485 ефективним у промислових умовах, де наявні електромагнітні перешкоди від електродвигунів, трансформаторів та іншого обладнання.

Інтерфейс RS-485 використовується для організації обміну даними між декількома пристроями, що підключені до спільної лінії зв'язку. Така структура називається багатоточковою топологією або шинною організацією. У межах однієї лінії може працювати декілька пристроїв, які обмінюються даними відповідно до визначених правил. Це дозволяє значно спростити побудову мережі та зменшити кількість необхідних кабельних з'єднань.

Програмна взаємодія через RS-485 здійснюється через послідовний порт, який є інтерфейсом між програмним застосунком і апаратним середовищем. Дані передаються у вигляді послідовного потоку бітів, що формуються відповідно до заданих параметрів зв'язку. До таких параметрів належать швидкість передавання, кількість бітів даних, наявність і тип контролю парності, а також кількість стоп-бітів. Узгодженість цих параметрів між пристроями є обов'язковою умовою коректного обміну даними.

Передавання даних у RS-485 зазвичай здійснюється у напівдуплексному режимі. Це означає, що у кожен момент часу дані можуть передаватися лише в одному напрямку. Для забезпечення такої роботи необхідно керувати режимом передавання і приймання, перемикаючи пристрій між цими станами. Це додає певної складності у програмній реалізації, оскільки необхідно враховувати час перемикання і уникати конфліктів на шині.

Важливим аспектом використання RS-485 є організація протоколу обміну даними. Сам по собі інтерфейс визначає лише фізичний рівень передавання, але не регламентує формат повідомлень або правила взаємодії. Тому для організації обміну використовуються протоколи верхнього рівня, які визначають структуру повідомлень, порядок передачі і правила обробки даних. Одним із найбільш поширених таких протоколів є Modbus.

Протокол Modbus визначає формат кадру, що містить адресу пристрою, функціональний код і дані. Це дозволяє організувати взаємодію між декількома пристроями у мережі, де кожен пристрій має унікальну адресу. Один із пристроїв виконує роль ведучого і ініціює обмін, а інші відповідають на запити. Такий підхід дозволяє уникнути конфліктів і забезпечити впорядкованість обміну даними.

При програмній реалізації взаємодії через RS-485 необхідно враховувати особливості роботи з послідовним портом. Програма повинна відкривати порт, встановлювати параметри зв'язку і виконувати операції читання та запису даних. Важливо забезпечити правильну синхронізацію операцій, оскільки дані можуть надходити з затримками або частинами. Для цього використовуються буфери і механізми очікування.

Ще одним важливим аспектом є контроль помилок. У процесі передавання даних можуть виникати спотворення, що призводять до некоректної інформації. Для виявлення таких ситуацій використовуються контрольні суми або інші механізми перевірки. У разі виявлення помилки програма повинна виконати повторну передачу або повідомити про збій.

Довжина лінії зв'язку і кількість підключених пристроїв також впливають на роботу системи. RS-485 дозволяє передавати дані на значні відстані, що може досягати сотень метрів, але при цьому необхідно враховувати зниження швидкості передачі та можливі втрати сигналу. Для забезпечення стабільної роботи використовуються термінуючі резистори, які зменшують відбиття сигналу і покращують якість зв'язку.

У сучасних інфокомунікаційних системах RS-485 часто використовується у поєднанні з мікроконтролерами та вбудованими системами. Це дозволяє створювати розподілені системи збору і обробки даних, де пристрої обмінюються інформацією у реальному часі. Програмна реалізація у таких системах повинна враховувати обмеження ресурсів і забезпечувати ефективну обробку даних.

Важливим є також питання сумісності. Оскільки RS-485 є стандартом фізичного рівня, різні пристрої можуть використовувати різні протоколи обміну. Тому при інтеграції систем необхідно забезпечити узгодженість форматів даних і правил взаємодії.

Таким чином, інтерфейс RS-485 є ефективним засобом організації обміну даними у промислових і інфокомунікаційних системах. Його переваги полягають у надійності, стійкості до завад і можливості побудови багатоточкових мереж. Програмна взаємодія через RS-485 потребує врахування особливостей послідовного передавання, організації протоколів обміну і забезпечення коректної обробки даних. Розуміння цих принципів є необхідним для розробки і експлуатації сучасних систем автоматизації і мережевих застосунків.

Практичне завдання

1. Налаштувати послідовний порт.
2. Реалізувати передавання даних через RS-485.
3. Реалізувати приймання даних.
4. Організувати обмін між двома пристроями.

5. Реалізувати простий протокол обміну.
6. Проаналізувати результати.
7. Дослідити вплив параметрів передачі.
8. Сформулювати висновки.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з принципами роботи послідовного обміну даними та особливостями інтерфейсу RS-485. Необхідно розуміти, що обмін даними здійснюється через послідовний порт, а коректність передачі залежить від правильного налаштування параметрів зв'язку та узгодженості цих параметрів між усіма пристроями.

На першому етапі необхідно визначити, який саме послідовний порт буде використовуватися у системі. Це може бути фізичний COM-порт або віртуальний порт, створений за допомогою відповідного програмного або апаратного забезпечення. Після цього потрібно відкрити порт у програмному застосунку і виконати його налаштування. Обов'язково слід задати такі параметри: швидкість передавання, кількість бітів даних, тип контролю парності та кількість стоп-бітів. Наприклад, типовими значеннями можуть бути 9600 біт/с, 8 біт даних, відсутність парності та 1 стоп-біт. Вибрані параметри необхідно зафіксувати.

На другому етапі необхідно реалізувати передавання даних через послідовний порт. Для цього потрібно сформувати повідомлення у вигляді послідовності байтів і передати його через відкритий порт. Повідомлення може бути простим текстовим рядком або структурованими даними. Важливо забезпечити правильне кодування даних і перевірити, що вони коректно передаються на інший пристрій або програмний модуль.

Далі необхідно реалізувати приймання даних. Програма повинна виконувати читання з послідовного порту і відображати отримані дані у консоль. Оскільки дані можуть надходити частинами, необхідно організувати читання у циклі та накопичення даних у буфері. Потрібно перевірити, що отримане повідомлення відповідає переданому.

На наступному етапі необхідно організувати двосторонній обмін даними. Один застосунок повинен виконувати роль передавача, а інший – приймача, після чого ролі можуть змінюватися. У випадку використання одного пристрою доцільно застосувати два взаємопов'язані віртуальні порти. Необхідно перевірити, що дані передаються і приймаються коректно в обох напрямках.

Особливу увагу слід приділити синхронізації обміну. Оскільки RS-485 часто працює у напівдуплексному режимі, необхідно забезпечити, щоб у кожен момент часу передавання здійснював лише один пристрій. Для цього потрібно реалізувати простий механізм керування обміном, наприклад використання затримок або визначеної послідовності передавання повідомлень.

Окремим завданням є визначення формату повідомлень. Необхідно обрати спосіб структурування даних, наприклад використання символу завершення повідомлення або передавання фіксованої довжини. Потрібно реалізувати обробку повідомлень відповідно до цього формату і перевірити правильність визначення їх меж.

Під час виконання роботи необхідно також передбачити обробку помилок. Зокрема, слід перевірити поведінку програми у випадку відсутності з'єднання, неправильних

параметрів порту або некоректних даних. Програма повинна виводити повідомлення про помилки і не завершувати роботу аварійно.

Після реалізації обміну даними необхідно провести серію експериментів. Доцільно змінювати швидкість передавання, розмір повідомлень і інтервали між передачами. Також варто перевірити роботу при передаванні довгих повідомлень і при створенні затримок. Усі результати необхідно зафіксувати.

На завершальному етапі необхідно виконати аналіз отриманих результатів. Потрібно оцінити стабільність передавання даних, вплив параметрів порту на якість зв'язку, а також ефективність обраного формату повідомлень. У висновках слід зазначити основні особливості роботи RS-485, можливі проблеми та способи їх усунення.

Таким чином, у процесі виконання роботи необхідно реалізувати обмін даними через RS-485 та дослідити особливості його функціонування, що є важливим для розуміння принципів програмної взаємодії з апаратними інтерфейсами.

Контрольні питання

1. Що таке RS-485?
2. Які його особливості?
3. Як здійснюється передавання даних?
4. Які параметри порту існують?
5. Що таке напівдуплекс?
6. Як організується обмін?
7. Які протоколи використовуються?
8. Які переваги RS-485?
9. Які проблеми можуть виникати?
10. Де використовується RS-485?

Зміст звіту

– назва роботи;
– мета роботи;
– опис інтерфейсу;
– реалізацію;
– результати;
– висновки.

ТЕМА 3. ПРОГРАМНІ ЗАСОБИ МОНІТОРИНГУ РОБОТИ ІНФОКОМУНІКАЦІЙНИХ СИСТЕМ

Практична робота 6. Перевірка доступності мережевих сервісів та ресурсів

Мета роботи – набути навичок перевірки доступності мережевих сервісів і аналізу параметрів їх роботи.

Ключові положення

Контроль доступності мережевих сервісів є однією з базових задач моніторингу інфокомунікаційних систем. У сучасних умовах більшість програмних систем функціонують у розподіленому середовищі, де взаємодія між компонентами відбувається через мережу. У таких системах навіть короткочасна недоступність сервісу може призвести до порушення роботи всієї системи, втрати даних або зниження якості обслуговування користувачів. Саме тому регулярна перевірка доступності є необхідною умовою забезпечення стабільної роботи.

Доступність сервісу визначається як його здатність приймати запити і повертати коректні відповіді у межах допустимого часу. Це поняття є складовою більш загальної характеристики надійності системи. Якщо сервіс не відповідає на запити або відповідає з великими затримками, це вважається порушенням доступності. У практиці розробки і експлуатації систем доступність часто вимірюється у відсотках часу, протягом якого сервіс працює без збоїв.

Для перевірки доступності використовуються різні програмні засоби і методи, що дозволяють оцінити стан мережевих ресурсів. Перевірка може виконуватися на різних рівнях моделі взаємодії. На мережевому рівні використовується запит типу ping, який базується на протоколі ICMP. Такий запит дозволяє визначити, чи доступний вузол у мережі, а також оцінити час проходження пакетів. Якщо вузол не відповідає на ping-запити, це може свідчити про проблеми з мережею або недоступність пристрою.

Однак перевірка на мережевому рівні не дає повної інформації про стан сервісу. Вузол може бути доступним, але сам сервіс, що працює на ньому, може бути недоступним або працювати некоректно. Тому важливим є контроль на прикладному рівні. Для цього використовуються HTTP-запити або інші протоколи, що дозволяють взаємодіяти безпосередньо з сервісом. Такий підхід дає змогу перевірити не лише факт доступності, але і правильність роботи застосунку.

Під час перевірки доступності аналізуються декілька ключових параметрів. Одним із них є час відповіді, який характеризує швидкість обробки запиту. Збільшення часу відповіді може свідчити про перевантаження системи або проблеми з мережею. Іншим важливим параметром є код статусу відповіді. Наприклад, для HTTP-запитів код 200 означає успішне виконання, тоді як коди 4xx і 5xx сигналізують про помилки. Аналіз цих кодів дозволяє визначити причини проблем.

Не менш важливим є показник стабільності роботи сервісу. Одиничні збої можуть бути випадковими, але регулярні відхилення свідчать про наявність системної проблеми. Тому перевірка доступності повинна виконуватися багаторазово з фіксованими інтервалами часу. Це дозволяє отримати повнішу картину стану сервісу і виявити тенденції у його роботі.

Для автоматизації перевірок використовуються спеціальні програмні сценарії або скрипти. Вони виконують запити до сервісів через задані інтервали часу, фіксують результати і за необхідності повідомляють про відхилення. Такий підхід дозволяє організувати безперервний моніторинг без участі користувача. Скрипти можуть бути реалізовані у різних мовах програмування і використовувати стандартні бібліотеки для роботи з мережею.

Важливою складовою автоматизованого контролю є журналювання результатів перевірок. Кожна перевірка повинна супроводжуватися записом, що містить час виконання, адресу сервісу, результат перевірки і, за можливості, час відповіді. Це дозволяє аналізувати історію роботи сервісу, визначати періоди недоступності і оцінювати загальний рівень надійності.

У процесі моніторингу необхідно враховувати можливість помилкових спрацювань. Наприклад, короткочасні збої у мережі можуть призвести до невдалого запиту, хоча сервіс у цілому працює коректно. Тому доцільно використовувати механізми повторних перевірок або аналізу декількох послідовних результатів перед формуванням висновку про недоступність.

Ще одним важливим аспектом є вибір інтервалу між перевірками. Часті перевірки дозволяють швидше виявляти проблеми, але можуть створювати додаткове навантаження на систему. Рідкісні перевірки зменшують навантаження, але можуть призвести до затримки у виявленні збоїв. Тому інтервал повинен підбиратися з урахуванням вимог до системи і її критичності.

У сучасних інфокомунікаційних системах контроль доступності часто є частиною більш комплексних систем моніторингу. Такі системи можуть поєднувати перевірку доступності з аналізом навантаження, використання ресурсів і журналів подій. Це дозволяє отримати повне уявлення про стан системи і своєчасно реагувати на проблеми.

Таким чином, контроль доступності мережевих сервісів є важливою складовою забезпечення надійності інфокомунікаційних систем. Він дозволяє своєчасно виявляти проблеми, аналізувати їх причини і забезпечувати стабільну роботу сервісів. Використання автоматизованих засобів контролю, аналіз параметрів роботи і ведення журналів є необхідними елементами ефективного моніторингу.

Практичне завдання

1. Реалізувати перевірку доступності вузла за допомогою ping.
2. Реалізувати перевірку вебсервісу через HTTP-запит.
3. Визначити час відповіді сервісу.
4. Проаналізувати коди статусу HTTP.
5. Реалізувати багаторазові перевірки через інтервали часу.
6. Зафіксувати результати перевірок.
7. Виявити можливі відхилення у роботі сервісу.
8. Сформулювати висновки.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з принципами перевірки доступності мережевих ресурсів та призначенням консольних утиліт, що використовуються

для діагностики мережі. Необхідно розуміти відмінність між перевіркою доступності вузла на мережевому рівні та перевіркою роботи сервісу на прикладному рівні.

На першому етапі необхідно виконати перевірку доступності вузла за допомогою утиліти ping. Для цього потрібно обрати мережевий ресурс, наприклад вебсервер або інший вузол у мережі, і виконати відповідну команду. У процесі виконання необхідно зафіксувати результати, зокрема наявність або відсутність відповіді, час відгуку та кількість втрачених пакетів. Отримані дані слід проаналізувати для визначення стану мережевого з'єднання.

На наступному етапі необхідно виконати перевірку доступності сервісу на прикладному рівні. Для цього слід використати консольну утиліту, що дозволяє формувати HTTP-запити. Потрібно виконати запит до вибраного вебсервісу та отримати відповідь. У результаті необхідно проаналізувати код статусу, вміст відповіді та заголовки. Особливу увагу слід приділити визначенню, чи відповідає сервіс коректно і чи повертає очікувані дані.

Далі необхідно організувати багаторазове виконання перевірок. Це можна реалізувати за допомогою повторного виконання команд або створення простого сценарію. Перевірки повинні виконуватися через однакові інтервали часу. Під час кожної перевірки необхідно фіксувати результати, зокрема час виконання, відповідь сервера та параметри з'єднання.

У процесі виконання роботи необхідно забезпечити фіксацію результатів перевірок. Для цього доцільно записувати отримані дані у текстовий файл або інший журнал. Кожен запис повинен містити часову мітку, адресу ресурсу та результат перевірки. Це дозволяє виконати подальший аналіз і оцінити стабільність роботи сервісу.

Особливу увагу слід приділити аналізу отриманих результатів. Необхідно визначити, чи спостерігаються затримки у відповіді, втрати пакетів або помилки у відповідях сервера. Також потрібно оцінити стабільність роботи сервісу протягом усього періоду спостереження.

Під час виконання роботи доцільно провести додаткові експерименти. Наприклад, змінити інтервал між перевірками, використати інші мережеві ресурси або перевірити недоступний вузол. Це дозволяє краще зрозуміти поведінку системи у різних умовах.

У завершенні необхідно узагальнити отримані результати, визначити ефективність використаних засобів перевірки та сформулювати висновки щодо доступності досліджуваних сервісів і особливостей їх роботи.

Контрольні питання

1. Що таке доступність сервісу?
2. Як виконується перевірка ping?
3. Що показує час відповіді?
4. Які коди статусу HTTP існують?
5. Як перевірити вебсервіс?
6. Чому важливий регулярний моніторинг?
7. Які параметри аналізуються?
8. Які проблеми можуть виникати?
9. Як автоматизувати перевірки?
10. Яке значення має стабільність сервісу?

Зміст звіту

– назва роботи;
– мета роботи;
– опис методів перевірки;
– результати експериментів;
– висновки.

Практична робота 7. Аналіз журналів подій інфокомунікаційних систем

Мета роботи – дослідити структуру журналів подій та набути навичок їх аналізу для виявлення помилок і збоїв у системі.

Ключові положення

Журнали подій є важливим елементом функціонування інфокомунікаційних систем, оскільки містять детальну інформацію про роботу програмних компонентів, виконані операції, стан системи та виникнення помилок. Вони формуються автоматично у процесі роботи системи і відображають послідовність подій, що відбуваються у часі. Завдяки цьому журнали є основним джерелом інформації для діагностики, аналізу збоїв та оцінки стабільності роботи сервісів.

Кожен запис журналу має визначену структуру. Як правило, він містить часову мітку, що фіксує момент виникнення події, рівень важливості, який характеризує тип події, а також текстовий опис. Часова мітка дозволяє відстежувати послідовність подій і аналізувати їх у динаміці. Рівень важливості може визначати, чи є подія інформаційною, попереджувальною або критичною. Опис події містить інформацію про дію, що була виконана, або про помилку, що виникла.

Журнали подій можуть формуватися на різних рівнях системи. Це можуть бути журнали операційної системи, журнали мережевих сервісів, журнали вебсерверів або окремих програмних застосунків. Кожен із цих журналів містить специфічну інформацію, яка використовується для аналізу відповідного компонента. Наприклад, журнали вебсерверів можуть містити інформацію про запити клієнтів, коди відповідей і помилки обробки.

Аналіз журналів подій є важливою задачею адміністрування і супроводу систем. Він може виконуватися вручну або автоматично. Ручний аналіз передбачає перегляд записів і пошук необхідної інформації. Такий підхід є ефективним для невеликих обсягів даних або для детального вивчення окремих ситуацій. Проте у великих системах обсяг журналів може бути значним, що ускладнює їх обробку.

Для обробки великих обсягів даних використовуються автоматизовані методи аналізу. До них належать фільтрація, пошук і кореляція подій. Фільтрація дозволяє відібрати записи за певними критеріями, наприклад за рівнем важливості або часовим інтервалом. Пошук дозволяє знаходити записи, що містять певні ключові слова або параметри. Кореляція подій дає змогу встановити взаємозв’язки між різними подіями і визначити причини виникнення проблем.

Одним із ключових завдань аналізу журналів є виявлення помилок і аномалій. Помилки можуть виникати у процесі виконання програм, обробки запитів або взаємодії з

іншими компонентами системи. Аномалії можуть проявлятися у вигляді нетипової поведінки, наприклад різкого збільшення кількості помилок або зміни характеру запитів. Виявлення таких ситуацій дозволяє своєчасно реагувати на проблеми і запобігати їх розвитку.

Аналіз журналів також використовується для дослідження продуктивності системи. За допомогою журналів можна визначити час виконання операцій, частоту запитів і навантаження на систему. Це дозволяє виявити вузькі місця і оптимізувати роботу сервісів.

Важливим аспектом є організація зберігання журналів. У системах із високим навантаженням обсяг журналів може швидко зростати, тому необхідно забезпечити їх ефективне зберігання і обробку. Для цього використовуються механізми архівації, ротації файлів і централізованого збору даних.

Окрему роль журнали подій відіграють у забезпеченні безпеки. Вони дозволяють виявляти спроби несанкціонованого доступу, підозрілі дії і атаки на систему. Аналіз журналів безпеки є важливим елементом захисту інформаційних систем.

У сучасних інфокомунікаційних системах журнали подій часто інтегруються з системами моніторингу і аналізу. Це дозволяє автоматично виявляти проблеми, формувати повідомлення і реагувати на інциденти у реальному часі. Такий підхід значно підвищує ефективність управління системою.

Таким чином, журнали подій є важливим джерелом інформації про стан і роботу інфокомунікаційних систем. Вони дозволяють виконувати діагностику, аналізувати помилки, оцінювати продуктивність і забезпечувати безпеку. Ефективний аналіз журналів базується на використанні методів фільтрації, пошуку і кореляції подій, що дозволяє своєчасно виявляти проблеми і підтримувати стабільну роботу системи.

Практичне завдання

1. Отримати журнал подій системи або сервісу.
2. Проаналізувати структуру записів.
3. Виконати фільтрацію за рівнем важливості.
4. Виявити помилки у журналі.
5. Проаналізувати послідовність подій.
6. Визначити причини збоїв.
7. Використати засоби пошуку.
8. Сформулювати висновки.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з призначенням журналів подій у інфокомунікаційних системах, їх структурою та основними підходами до аналізу. Необхідно розуміти, які типи подій фіксуються у журналах, які рівні важливості використовуються та яку інформацію містять записи.

На першому етапі необхідно отримати журнал подій із системи або мережевого сервісу. Це може бути журнал операційної системи, вебсервера, застосунку або іншого програмного компонента. Потрібно визначити місце зберігання журналу, відкрити його за допомогою відповідних засобів та підготувати до аналізу. Доцільно використати консольні утиліти або текстові редактори, що дозволяють переглядати великі файли.

Далі необхідно проаналізувати структуру записів журналу. Потрібно визначити, які елементи містить кожен запис, зокрема часову мітку, рівень важливості та опис події. Необхідно встановити формат представлення дати і часу, спосіб позначення рівнів важливості та структуру текстового повідомлення. Результати аналізу структури слід зафіксувати.

На наступному етапі необхідно виконати фільтрацію записів журналу. Потрібно відібрати записи за певними критеріями, наприклад за рівнем важливості або за наявністю ключових слів. Особливу увагу слід приділити записам, що містять інформацію про помилки або попередження. Для цього доцільно використати засоби пошуку або фільтрації.

Після цього необхідно виконати пошук і аналіз помилок. Потрібно визначити, які саме помилки виникали, як часто вони повторюються та за яких умов з'являються. Необхідно зафіксувати знайдені помилки і проаналізувати їх можливі причини.

Особливу увагу слід приділити аналізу послідовності подій. Необхідно розглянути записи у хронологічному порядку та визначити, які події передували виникненню помилок. Це дозволяє встановити причинно-наслідкові зв'язки і краще зрозуміти поведінку системи.

Додатково доцільно проаналізувати частоту появи подій і визначити, чи є повторювані або аномальні ситуації. Це дозволяє оцінити стабільність роботи системи та виявити потенційні проблеми.

Під час виконання роботи необхідно зафіксувати результати аналізу, зокрема приклади записів, виявлені помилки та їх характеристики. Усі дані слід систематизувати для подальшого узагальнення.

На завершальному етапі необхідно виконати узагальнення результатів. Потрібно оцінити інформативність журналу подій, ефективність методів аналізу та можливість використання журналів для діагностики системи. У висновках слід зазначити основні типи виявлених подій, причини помилок та значення журналів подій для забезпечення стабільної роботи інфокомунікаційних систем.

Контрольні питання

1. Що таке журнал подій?
2. Яка структура запису?
3. Які рівні важливості існують?
4. Як виконується аналіз журналів?
5. Що таке фільтрація?
6. Як виявляються помилки?
7. Що таке кореляція подій?
8. Які інструменти використовуються?
9. Чому важливий аналіз журналів?
10. Які проблеми можуть виникати?

Зміст звіту

– назва роботи;
– мета роботи;
– опис журналів;
– результати аналізу;
– висновки.

Практична робота 8. Отримання інформації про стан мережевих пристроїв за допомогою SNMP

Мета роботи – ознайомитися з принципами роботи протоколу SNMP та отримання інформації про стан мережевих пристроїв.

Ключові положення

Протокол SNMP (Simple Network Management Protocol) є стандартним засобом для моніторингу та управління мережевими пристроями в інфокомунікаційних системах. Він широко використовується для отримання інформації про стан обладнання, такого як маршрутизатори, комутатори, сервери та інші мережеві вузли. SNMP дозволяє централізовано збирати дані про роботу пристроїв і контролювати їх функціонування у реальному часі.

Основою роботи SNMP є модель взаємодії між менеджером і агентом. Менеджер є програмним компонентом, що виконує функції збору та аналізу інформації. Агент, у свою чергу, є програмним модулем, що працює на мережевому пристрої і надає доступ до його параметрів. Менеджер надсилає запити агенту, а агент повертає відповідні значення. Такий підхід дозволяє організувати централізований контроль за великою кількістю пристроїв.

Обмін даними у SNMP здійснюється за допомогою спеціальних повідомлень, які передаються через мережу. Основними операціями є отримання значення параметра, встановлення значення та отримання групи параметрів. Найчастіше використовується операція отримання, яка дозволяє дізнатися поточний стан пристрою. Крім того, агенти можуть самостійно надсилати повідомлення про події, що дозволяє оперативно реагувати на зміни у системі.

Дані у SNMP організовані у вигляді бази керуючої інформації. Ця база містить набір параметрів, що описують стан пристрою, його інтерфейси, навантаження, кількість переданих і отриманих пакетів та інші характеристики. Кожен параметр має унікальний ідентифікатор, за допомогою якого до нього можна звертатися. Така організація дозволяє уніфікувати доступ до різних типів пристроїв.

SNMP дозволяє отримувати широкий спектр інформації про стан мережевих пристроїв. Це може бути інформація про мережеві інтерфейси, їх стан, швидкість передачі, кількість помилок, обсяг трафіку, використання ресурсів пристрою та інші параметри. Аналіз цих даних дозволяє оцінити продуктивність системи, виявити перевантаження і своєчасно реагувати на проблеми.

Важливим аспектом використання SNMP є періодичність збору даних. Регулярне отримання інформації дозволяє відстежувати зміни у стані системи і виявляти тенденції. Це є основою для побудови систем моніторингу, які забезпечують безперервний контроль за роботою мережі.

Протокол SNMP має декілька версій, що відрізняються рівнем безпеки і функціональними можливостями. Ранні версії забезпечують базову функціональність, тоді як новіші версії додають механізми автентифікації та захисту даних. Використання захищених версій є важливим для запобігання несанкціонованому доступу до інформації.

У практиці SNMP часто використовується разом із іншими засобами моніторингу. Це дозволяє поєднувати різні джерела інформації і отримувати більш повне уявлення про стан

системи. Наприклад, дані SNMP можуть доповнюватися журналами подій або результатами перевірки доступності.

Таким чином, протокол SNMP є ефективним інструментом для отримання інформації про стан мережевих пристроїв. Він забезпечує централізований збір даних, дозволяє контролювати роботу системи і своєчасно виявляти проблеми. Розуміння принципів роботи SNMP є необхідним для організації ефективного моніторингу інфокомунікаційних систем.

Практичне завдання

1. Налаштувати доступ до SNMP-пристрою.
2. Виконати запит параметрів.
3. Отримати інформацію про інтерфейси.
4. Проаналізувати отримані дані.
5. Виконати декілька запитів.
6. Зафіксувати результати.
7. Проаналізувати стан пристрою.
8. Сформулювати висновки.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно ознайомитися з принципами роботи протоколу SNMP, його архітектурою та основними поняттями, такими як менеджер, агент і база керуючої інформації. Необхідно розуміти, що обмін даними відбувається шляхом надсилання запитів до агента і отримання значень параметрів, які характеризують стан мережевого пристрою.

На першому етапі необхідно підготувати середовище для роботи з SNMP. Потрібно обрати мережевий пристрій або програмний агент, який підтримує SNMP. Це може бути реальний мережевий пристрій або програмний емулятор. Далі необхідно налаштувати доступ до агента. Для цього слід визначити адресу пристрою, номер порту та параметри доступу. У разі використання спрощених налаштувань необхідно вказати рядок доступу, який використовується для автентифікації запитів. Усі параметри необхідно зафіксувати.

На наступному етапі необхідно використати консольні утиліти або програмні засоби для виконання SNMP-запитів. Потрібно виконати запит для отримання окремого параметра, використовуючи його ідентифікатор. Отримане значення необхідно вивести у консоль і зафіксувати. Далі доцільно виконати запити для отримання декількох параметрів, що дозволяє оцінити стан різних компонентів пристрою.

Після цього необхідно виконати більш детальний збір даних. Потрібно отримати інформацію про мережеві інтерфейси, їх стан, а також параметри, що характеризують обсяг переданих і отриманих даних. У процесі виконання запитів необхідно звернути увагу на формат отриманих даних і спосіб їх представлення.

Особливу увагу слід приділити правильності виконання запитів. У разі помилок необхідно перевірити параметри доступу, правильність ідентифікаторів параметрів і доступність пристрою. Програма або утиліта повинна коректно повідомляти про помилки і дозволяти їх аналізувати.

Після отримання даних необхідно виконати їх аналіз. Потрібно визначити, які параметри характеризують стан пристрою, оцінити значення показників і зробити висновки

щодо його роботи. Доцільно порівняти значення різних параметрів і визначити, чи відповідають вони очікуваним.

У процесі виконання роботи рекомендується провести декілька запитів через різні проміжки часу. Це дозволяє оцінити динаміку змін параметрів і визначити, як змінюється стан пристрою у часі. Отримані результати необхідно зафіксувати для подальшого аналізу.

На завершальному етапі необхідно узагальнити результати виконання роботи. Потрібно оцінити можливості протоколу SNMP для отримання інформації, визначити його переваги і обмеження, а також сформулювати висновки щодо доцільності його використання у задачах моніторингу інфокомунікаційних систем.

Контрольні питання

1. Що таке SNMP?
2. Які компоненти використовуються?
3. Що таке MIB?
4. Як виконуються запити?
5. Які дані можна отримати?
6. Які версії SNMP існують?
7. Які переваги протоколу?
8. Які обмеження?
9. Як забезпечується безпека?
10. Де використовується SNMP?

Зміст звіту

– назва роботи;
– мета роботи;
– опис SNMP;
– результати роботи;
– висновки.

ТЕМА 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ УПРАВЛІННЯ ІНФОКОМУНІКАЦІЙНИМИ СИСТЕМАМИ

Практична робота 9. Створення скрипта автоматичного контролю доступності мережевих сервісів

Мета роботи – набути навичок створення скриптів автоматичного контролю доступності мережевих сервісів і аналізу результатів моніторингу.

Ключові положення

Автоматичний контроль доступності мережевих сервісів є важливою складовою супроводу інфокомунікаційних систем. У сучасних умовах програмні сервіси повинні працювати безперервно, а будь-яке порушення їх доступності має бути виявлене якомога раніше. Для цього використовуються програмні засоби, що виконують регулярні перевірки стану вузлів, серверів і мережевих застосунків.

Доступність сервісу означає його здатність приймати запити і повертати коректні відповіді у межах допустимого часу. Якщо сервіс не відповідає, повертає помилковий код стану або час відповіді перевищує встановлене значення, це може свідчити про наявність проблем у мережі, на сервері або у самому застосунку. Саме тому автоматизований контроль повинен враховувати не лише факт відповіді, але і її параметри.

Одним із найпростіших підходів до контролю є періодичне виконання перевірок за допомогою мережевих запитів. Для цього можуть використовуватися ICMP-запити, HTTP-запити, перевірка доступності порту або спроба встановлення мережевого з’єднання. Вибір методу залежить від типу ресурсу, який контролюється. Якщо йдеться про вебсервіс, доцільно використовувати HTTP-запит, оскільки він дає змогу оцінити не лише доступність вузла, але і стан прикладного сервісу.

Скрипт автоматичного контролю є програмою, яка без участі користувача виконує задану послідовність дій: надсилає запит до сервісу, фіксує результат, визначає наявність або відсутність проблеми, а за необхідності – записує інформацію до журналу або виводить попередження. Такий підхід дозволяє проводити багаторазові перевірки через однакові проміжки часу і накопичувати результати спостереження.

Однією з важливих задач такого скрипта є фіксація часу перевірки. Кожен запис результату повинен містити часову мітку, адресу сервісу, тип перевірки, результат виконання та за можливості час відповіді. Це дає змогу не лише спостерігати поточний стан, але й аналізувати зміну доступності у часі.

Автоматизований контроль часто поєднується з журналюванням. Якщо результати перевірок зберігаються у текстовому файлі або іншому сховищі, надалі можна визначати періоди недоступності сервісу, частоту виникнення збоїв і загальний рівень стабільності. Це є основою для подальшого аналізу і прийняття рішень щодо покращення інфраструктури.

У простих випадках скрипт може лише виводити повідомлення про стан сервісу у консоль. У більш розвиненому варіанті він може записувати інформацію у файл, формувати окремі повідомлення про помилки або виконувати додаткові дії при виявленні збою. Наприклад, якщо сервіс недоступний, скрипт може зафіксувати це як окрему подію або сформувати сигнал для адміністратора.

Під час розробки такого скрипта необхідно враховувати можливість помилок з’єднання, тайм-аутів і нестабільної роботи мережі. Одноразова невдала перевірка не завжди означає реальну відмову сервісу, тому доцільно передбачати повторну перевірку або враховувати декілька послідовних невдалих спроб. Це дозволяє зменшити кількість помилкових спрацювань.

Ще одним важливим питанням є вибір інтервалу між перевірками. Якщо інтервал буде надто малим, скрипт може створювати зайве навантаження на мережу або сервіс. Якщо надто великим, збої можуть виявлятися із запізненням. Тому інтервал підбирається залежно від критичності сервісу і вимог до оперативності контролю.

Практична реалізація такого скрипта дає змогу поєднати знання з мережевого програмування, роботи з HTTP, обробки помилок, циклічного виконання команд і журналювання результатів. Це робить роботу важливою з точки зору формування навичок автоматизації моніторингу інфокомунікаційних систем.

Таким чином, створення скрипта автоматичного контролю доступності мережевих сервісів є важливою задачею практичної підготовки. Воно дає змогу засвоїти принципи автоматизованого моніторингу, навчитися виконувати регулярні перевірки і фіксувати результати спостереження у програмний спосіб.

Практичне завдання

1. Обрати мережевий сервіс або вебресурс для контролю доступності.
2. Реалізувати скрипт, що виконує перевірку доступності сервісу.
3. Забезпечити багаторазове виконання перевірок через заданий інтервал часу.
4. Реалізувати визначення результату перевірки на основі відповіді сервісу.
5. Додати фіксацію часу виконання кожної перевірки.
6. Реалізувати виведення результатів у консоль.
7. Забезпечити запис результатів перевірок до текстового журналу.
8. Реалізувати обробку помилок з’єднання або перевищення часу очікування.
9. Проаналізувати отримані результати спостереження.
10. Сформулювати висновки щодо стабільності роботи перевірюваного сервісу.

Методичні вказівки до виконання практичного завдання

Перед виконанням роботи необхідно визначити, який саме сервіс буде контролюватися. Доцільно обрати вебсервіс або мережевий ресурс, доступність якого можна перевірити за допомогою HTTP-запиту або іншого стандартного способу. Якщо використовується вебсервіс, слід заздалегідь визначити, який код відповіді вважатиметься ознакою нормальної роботи.

На першому етапі потрібно реалізувати найпростіший варіант скрипта, який виконує одну перевірку доступності сервісу. Скрипт повинен звернутися до ресурсу, отримати відповідь і визначити, чи є сервіс доступним. Результат доцільно подати у зрозумілому текстовому вигляді.

На наступному етапі необхідно додати циклічне виконання перевірок через однакові проміжки часу. Для цього можна використати цикл із затримкою між ітераціями. Під час кожної перевірки слід фіксувати поточний час, щоб результати можна було аналізувати у хронологічному порядку.

Далі потрібно реалізувати журналювання результатів. Кожна перевірка повинна записуватися до текстового файлу. У записі доцільно вказувати дату і час, адресу сервісу, результат перевірки та, за можливості, час відповіді. Якщо виникла помилка з'єднання або перевищено час очікування, це також слід явно фіксувати.

Особливу увагу необхідно приділити обробці помилок. Скрипт не повинен аварійно завершуватися у разі недоступності сервісу. Натомість він має коректно зафіксувати помилку і продовжити роботу. Для цього слід передбачити перехоплення виключень або перевірку кодів помилок.

Після завершення реалізації необхідно провести серію перевірок і проаналізувати журнал. За результатами спостереження потрібно визначити, чи були випадки недоступності, які коди відповідей отримувалися, чи змінювався час відповіді та наскільки стабільно працював сервіс.

У висновках слід оцінити ефективність реалізованого скрипта, зручність автоматизованого контролю та доцільність використання такого підходу у задачах моніторингу інфокомунікаційних систем. Також варто зазначити, які можливості можна було б додати у подальшому, наприклад формування попереджень або перевірку декількох сервісів одночасно.

Під час виконання роботи необхідно дотримуватися принципів академічної доброчесності, самостійно виконувати програмну реалізацію та коректно оформлювати використані джерела інформації.

Контрольні питання

1. Що розуміють під доступністю мережевого сервісу?
2. Які способи перевірки доступності мережевих ресурсів існують?
3. Чому для вебсервісів доцільно використовувати HTTP-запити?
4. Які дані доцільно фіксувати під час автоматичної перевірки сервісу?
5. Для чого потрібна часова мітка у журналі перевірок?
6. Що таке тайм-аут і чому його необхідно враховувати?
7. Які причини можуть призводити до недоступності сервісу?
8. Чому важливо реалізувати обробку помилок у скрипті контролю?
9. Яке призначення журналу результатів перевірки?
10. Від чого залежить вибір інтервалу між перевірками?
11. Чому одноразова невдала перевірка не завжди свідчить про відмову сервісу?
12. Які можливості розширення може мати скрипт автоматичного контролю?

Зміст звіту

– назва практичної роботи;
– мета виконання практичної роботи;
– короткі теоретичні відомості про контроль доступності мережевих сервісів;
– опис обраного сервісу або ресурсу для перевірки;
– опис алгоритму роботи скрипта;
– текст або фрагменти реалізованого скрипта;
– результати виконання перевірок;
– приклади записів журналу;

– аналіз отриманих результатів;
– висновки за результатами виконання практичної роботи;
– перелік використаних джерел інформації.

Практична робота 10. Використання консольних утиліт для управління мережевими сервісами

Мета роботи – ознайомитися з консольними утилітами адміністрування та набути навичок управління мережевими сервісами за допомогою командного рядка.

Ключові положення

Консольні утиліти є важливим інструментом адміністрування інфокомунікаційних систем і мережевих сервісів. Вони дозволяють виконувати діагностику, налаштування і управління ресурсами без використання графічного інтерфейсу. Такий підхід забезпечує високу швидкість роботи, точність виконання команд і можливість автоматизації.

Утиліти командного рядка дають змогу взаємодіяти безпосередньо з операційною системою і мережевим стеком. Вони використовуються для перевірки доступності вузлів, аналізу маршрутів передачі даних, перегляду мережевих параметрів і управління сервісами. Це дозволяє отримати детальну інформацію про стан системи і виконувати необхідні дії для її підтримки.

До базових утиліт належать засоби перевірки доступності, які дозволяють визначити, чи відповідає мережевий вузол на запити. Також використовуються утиліти для аналізу маршруту, що дає змогу встановити шлях проходження пакетів у мережі. Це важливо для діагностики проблем з’єднання.

Окрему групу становлять утиліти для перегляду параметрів мережевих інтерфейсів. Вони дозволяють отримати інформацію про IP-адреси, стан інтерфейсів і конфігурацію мережі. Це необхідно для налаштування і контролю роботи системи.

Важливими є також утиліти для роботи з мережевими з’єднаннями і портами. Вони дозволяють визначити, які сервіси працюють у системі, які порти відкриті і які з’єднання встановлені. Це дає змогу контролювати роботу сервісів і забезпечувати безпеку.

Для взаємодії з вебсервісами використовуються утиліти, що дозволяють формувати HTTP-запити і отримувати відповіді. Це дає змогу тестувати API, перевіряти роботу серверів і аналізувати результати.

Консольні утиліти також використовуються для управління сервісами операційної системи. Вони дозволяють запускати, зупиняти і перезапускати служби, а також перевіряти їх стан. Це є важливим для підтримки працездатності мережевих застосунків.

Однією з переваг консольних утиліт є можливість їх використання у сценаріях автоматизації. Команди можуть об’єднуватися у скрипти, що дозволяє виконувати складні операції без участі користувача. Це підвищує ефективність роботи і зменшує ймовірність помилок.

Під час використання таких утиліт необхідно враховувати правильність введення команд і параметрів. Неправильне використання може призвести до некоректної роботи системи або втрати даних. Тому важливо розуміти призначення кожної команди і її параметрів.

Таким чином, консольні утиліти є ефективним засобом управління мережевими сервісами, що дозволяє виконувати широкий спектр задач від діагностики до конфігурування і контролю роботи системи.

Практичне завдання

1. Виконати перевірку доступності вузла за допомогою консольної утиліти.
2. Визначити маршрут передачі пакетів до заданого вузла.
3. Отримати інформацію про мережеві інтерфейси системи.
4. Визначити відкриті порти і активні з’єднання.
5. Виконати HTTP-запит до вебсервісу через консольну утиліту.
6. Проаналізувати отриману відповідь.
7. Перевірити стан одного з мережевих сервісів.
8. Виконати запуск або перезапуск сервісу.
9. Зафіксувати результати виконаних команд.
10. Сформулювати висновки щодо використання консольних утиліт.

Методичні вказівки до виконання практичного завдання

Перед виконанням роботи необхідно ознайомитися з основними консольними утилітами для роботи з мережею і сервісами. Слід визначити їх призначення і параметри.

На першому етапі потрібно виконати перевірку доступності вузла. Це дозволяє оцінити стан мережевого з’єднання.

Далі необхідно визначити маршрут передачі пакетів. Це дає змогу виявити можливі проблеми у мережі.

Після цього слід отримати інформацію про мережеві інтерфейси і активні з’єднання. Це дозволяє оцінити конфігурацію системи.

Наступним етапом є виконання HTTP-запиту і аналіз відповіді. Це дозволяє перевірити роботу вебсервісу.

Після цього потрібно виконати управління сервісом, наприклад його перезапуск. Це дозволяє перевірити можливості адміністрування.

Усі результати необхідно зафіксувати і проаналізувати.

Контрольні питання

1. Що таке консольні утиліти?
2. Які задачі вони дозволяють виконувати?
3. Як перевіряється доступність вузла?
4. Що показує маршрут передачі пакетів?
5. Як отримати інформацію про мережеві інтерфейси?
6. Як визначити відкриті порти?
7. Як виконати HTTP-запит через консоль?
8. Як управляти сервісами?
9. Які переваги консольних утиліт?
10. Які ризики їх використання?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис використаних утиліт;
– результати виконання команд;
– аналіз отриманих даних;
– висновки.

Практична робота 11. Налаштування та дослідження роботи серверних служб

Мета роботи – набути навичок налаштування серверних служб та дослідження їх роботи у різних умовах.

Ключові положення

Серверні служби є основними програмними компонентами інфокомунікаційних систем, що забезпечують обробку запитів, надання ресурсів і взаємодію між клієнтами та іншими сервісами. Вони функціонують у фоновому режимі та забезпечують безперервну роботу системи незалежно від наявності активного користувача. Серверні служби є основою більшості сучасних інформаційних систем, оскільки саме вони реалізують логіку обробки даних, зберігання інформації та забезпечення доступу до ресурсів.

До найбільш поширених серверних служб належать вебсервери, що обробляють HTTP-запити, сервери баз даних, які відповідають за зберігання та обробку інформації, файлові сервери, що забезпечують доступ до файлів, а також служби обміну повідомленнями, які реалізують взаємодію між компонентами системи. Кожен із цих типів серверів виконує специфічні функції, але всі вони працюють за спільним принципом: приймання запитів, їх обробка і формування відповіді.

Налаштування серверних служб полягає у визначенні параметрів їх роботи, які безпосередньо впливають на функціонування системи. До таких параметрів належать номер порту, на якому працює сервіс, адреса прив’язки, правила доступу, шляхи до ресурсів, параметри обробки запитів та інші конфігураційні характеристики. Ці параметри зберігаються у конфігураційних файлах або системах керування конфігураціями і визначають поведінку сервісу у різних умовах.

Процес налаштування містить декілька етапів. Спочатку виконується вибір значень параметрів відповідно до вимог системи. Далі необхідно перевірити коректність введених даних і узгодженість параметрів між собою. Після цього зміни застосовуються до сервісу, що зазвичай потребує його перезапуску. Неправильна конфігурація може призвести до недоступності сервісу, зниження продуктивності або порушення безпеки, тому перевірка параметрів є обов’язковою.

Дослідження роботи серверних служб передбачає аналіз їх функціонування у різних умовах. Це містить перевірку доступності сервісу, оцінку часу відповіді на запити, аналіз навантаження та виявлення можливих помилок. Такий підхід дозволяє оцінити ефективність роботи сервісу і визначити, наскільки він відповідає вимогам до продуктивності та надійності.

Одним із важливих параметрів є час відповіді сервера. Він характеризує швидкість обробки запитів і є критичним для користувацького досвіду. Збільшення часу відповіді може свідчити про перевантаження системи, неефективну конфігурацію або проблеми у програмній реалізації. Аналіз цього параметра дозволяє виявити вузькі місця і визначити напрямки оптимізації.

Не менш важливим є аналіз навантаження на сервер. Він дозволяє оцінити використання ресурсів, таких як процесор, пам’ять і мережеві інтерфейси. Надмірне навантаження може призвести до зниження продуктивності або відмови сервісу, тому необхідно забезпечити баланс між навантаженням і доступними ресурсами.

Важливим елементом є управління станом сервісу. Серверна служба може бути запущена, зупинена або перезапущена. Запуск сервісу переводить його у робочий стан, зупинка припиняє обробку запитів, а перезапуск дозволяє застосувати нові налаштування або відновити роботу після збою. Контроль цих станів є необхідним для забезпечення стабільної роботи системи.

У процесі дослідження доцільно використовувати журнали подій і засоби моніторингу. Журнали містять інформацію про запити, помилки і внутрішні процеси сервісу, що дозволяє аналізувати його роботу у часі. Засоби моніторингу дають змогу відстежувати параметри роботи у реальному часі і швидко реагувати на відхилення.

Окрему увагу слід приділити безпеці серверних служб. Налаштування повинно містити обмеження доступу, контроль прав користувачів і використання захищених протоколів. Неправильні налаштування безпеки можуть призвести до несанкціонованого доступу і втрати даних.

У сучасних інфокомунікаційних системах серверні служби часто працюють у розподіленому середовищі, де декілька серверів взаємодіють між собою. У таких системах важливим є узгодження налаштувань і забезпечення стабільної взаємодії між компонентами. Це може містити балансування навантаження, резервування ресурсів і автоматичне відновлення роботи у разі збоїв.

Таким чином, налаштування і дослідження серверних служб є важливими задачами адміністрування, що забезпечують ефективну і стабільну роботу інфокомунікаційних систем. Вони містять визначення параметрів роботи, контроль стану сервісу, аналіз продуктивності і використання засобів моніторингу. Розуміння цих процесів є необхідним для забезпечення надійності і ефективності сучасних мережевих систем.

Практичне завдання

1. Обрати серверну службу для дослідження (вебсервер або інший сервіс).
2. Ознайомитися зі структурою конфігураційних файлів.
3. Виконати базове налаштування сервісу (порт, адреса, параметри доступу).
4. Запустити серверну службу.
5. Перевірити доступність сервісу з клієнтського застосунку або браузера.
6. Змінити один або декілька параметрів конфігурації.
7. Перезапустити сервіс і перевірити зміни у роботі.
8. Проаналізувати журнали подій сервісу.
9. Дослідити вплив змін конфігурації на роботу сервісу.
10. Сформулювати висновки.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно обрати серверну службу, яка буде досліджуватися. Доцільно використати вебсервер або інший доступний сервіс, що підтримує налаштування через конфігураційні файли. Необхідно переконатися, що обраний сервіс встановлений у системі та доступний для запуску і налаштування.

На першому етапі необхідно знайти та відкрити конфігураційний файл серверної служби. Потрібно ознайомитися зі структурою файлу, визначити його формат і призначення окремих параметрів. Особливу увагу слід приділити таким параметрам, як номер порту, адреса прив'язки, шляхи до ресурсів, правила доступу та інші налаштування, що впливають на роботу сервісу. Виявлені параметри необхідно зафіксувати.

Далі необхідно виконати базове налаштування сервісу. Потрібно встановити значення основних параметрів відповідно до умов виконання роботи. Після внесення змін необхідно зберегти конфігураційний файл і виконати запуск серверної служби. У разі необхідності слід використати консольні утиліти для керування сервісом.

Після запуску необхідно перевірити доступність сервісу. Для цього потрібно використати клієнтський застосунок, браузер або консольні утиліти. Необхідно виконати запит до сервера і переконатися, що він відповідає коректно. Результати перевірки слід зафіксувати.

На наступному етапі необхідно змінити один або декілька параметрів конфігурації. Наприклад, можна змінити порт, адресу прив'язки або параметри доступу. Після внесення змін необхідно зберегти конфігурацію і виконати перезапуск сервісу. Далі слід повторно перевірити доступність сервісу і визначити, як внесені зміни вплинули на його роботу.

Особливу увагу необхідно приділити аналізу журналів подій. Потрібно знайти файл журналу і проаналізувати записи, що стосуються запуску сервісу, обробки запитів і можливих помилок. Необхідно визначити, чи виникали помилки під час запуску або роботи сервісу, а також зафіксувати відповідні записи.

У процесі виконання роботи доцільно провести додаткові експерименти. Наприклад, можна змінити декілька параметрів одночасно, перевірити роботу сервісу при різних налаштуваннях або проаналізувати поведінку при некоректній конфігурації. Це дозволяє краще зрозуміти вплив параметрів на роботу системи.

Після виконання усіх етапів необхідно провести аналіз отриманих результатів. Потрібно оцінити, як зміна конфігурації впливає на доступність, продуктивність і стабільність роботи сервісу. Також слід визначити, які параметри є найбільш критичними для його функціонування.

У завершенні необхідно сформулювати висновки щодо впливу конфігурації на роботу серверної служби, визначити основні закономірності та зробити узагальнення щодо процесу налаштування і дослідження серверних сервісів у інфокомунікаційних системах.

Контрольні питання

1. Що таке серверна служба?
2. Які основні типи серверних служб існують?
3. Що містить конфігураційний файл сервісу?
4. Які параметри впливають на роботу сервера?
5. Як виконується запуск і зупинка сервісу?

6. Що таке перезапуск сервісу?
7. Як перевірити доступність серверної служби?
8. Яку роль відіграють журнали подій?
9. Які помилки можуть виникати при налаштуванні?
10. Чому важливо тестувати зміни конфігурації?

Зміст звіту

– назву практичної роботи;
– мету виконання практичної роботи;
– опис обраної серверної служби;
– характеристику конфігураційних параметрів;
– опис виконаних налаштувань;
– результати перевірки роботи сервісу;
– аналіз журналів подій;
– висновки за результатами виконання практичної роботи;
– перелік використаних джерел інформації.

ТЕМА 5. ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ ПРОГРАМ ТА ДАНИХ В ІНФОКОМУНІКАЦІЙНИХ СИСТЕМАХ

Практична робота 12. Дослідження механізмів аутентифікації та авторизації у мережевих сервісах

Мета роботи – дослідити механізми аутентифікації та авторизації у мережевих сервісах і принципи контролю доступу.

Ключові положення

Аутентифікація та авторизація є базовими механізмами забезпечення безпеки у мережевих сервісах і програмних системах, що взаємодіють через мережу. Вони визначають, хто отримує доступ до системи, а також які дії дозволені конкретному користувачу або програмному компоненту. Ці механізми реалізуються на рівні застосунків і є невід'ємною частиною сучасних вебсервісів, API та розподілених систем.

Аутентифікація є процесом перевірки особи користувача або сервісу. Її основною метою є підтвердження того, що суб'єкт, який намагається отримати доступ, є тим, за кого себе видає. Найпоширенішим методом аутентифікації є використання логіна і пароля. У цьому випадку облікові дані передаються у запиті до сервера, де перевіряються на відповідність збереженим значенням. У разі успішної перевірки доступ до системи надається.

Окрім класичного підходу з використанням логіна і пароля, у сучасних системах широко застосовуються токени доступу. Після успішної аутентифікації сервер генерує токен, який передається клієнту і використовується для подальших запитів. Такий підхід дозволяє уникнути повторного введення облікових даних і спрощує взаємодію між клієнтом і сервером. Токени можуть мати обмежений час дії, що підвищує рівень безпеки.

Авторизація є наступним етапом після аутентифікації і визначає права доступу користувача до ресурсів системи. Вона базується на політиках доступу, які визначають, які дії дозволені для конкретного користувача або групи користувачів. Найчастіше використовується модель ролей, де кожному користувачу призначається певна роль, а ролі визначають набір дозволених дій.

У процесі авторизації система перевіряє, чи має користувач право виконати певну операцію. Наприклад, один користувач може мати доступ лише до перегляду даних, тоді як інший – до їх редагування або видалення. Такий підхід дозволяє обмежити доступ до критичних ресурсів і забезпечити захист інформації.

У мережевих сервісах механізми аутентифікації та авторизації часто реалізуються через HTTP-запити. Дані для аутентифікації можуть передаватися у заголовках або у тілі запиту. Наприклад, облікові дані можуть передаватися у вигляді закодованого рядка, а токени – у спеціальних заголовках. Сервер обробляє ці дані і приймає рішення щодо надання доступу.

Важливим аспектом є забезпечення безпеки передавання даних. Дані аутентифікації, такі як логіни, паролі або токени, повинні передаватися у зашифрованому вигляді. Для цього використовується захищений протокол HTTPS, який забезпечує конфіденційність і цілісність

даних. Використання незахищених каналів може призвести до перехоплення облікових даних і несанкціонованого доступу.

Окрім шифрування, необхідно враховувати й інші аспекти безпеки. Зокрема, важливо обмежувати кількість спроб входу, використовувати механізми блокування облікових записів у разі підозрілої активності, а також контролювати час дії токенів. Це дозволяє зменшити ризик атак і підвищити загальний рівень захисту системи.

У сучасних системах часто застосовуються багатофакторні методи аутентифікації, які передбачають використання декількох незалежних факторів перевірки. Це може бути поєднання пароля, одноразового коду або апаратного пристрою. Такий підхід значно підвищує рівень безпеки.

Дослідження механізмів аутентифікації та авторизації дозволяє зрозуміти принципи захисту мережевих сервісів і оцінити їх ефективність у різних умовах. Аналіз цих механізмів дає змогу визначити можливі вразливості, оцінити рівень захисту і сформувати рекомендації щодо покращення безпеки.

Таким чином, аутентифікація та авторизація є ключовими компонентами систем безпеки, що забезпечують контроль доступу до ресурсів і захист інформації. Вони базуються на перевірці особи користувача і визначенні його прав, використовують сучасні технології передавання даних і потребують належної реалізації для забезпечення надійності інфокомунікаційних систем.

Практичне завдання

1. Обрати вебсервіс або API, що підтримує аутентифікацію.
2. Виконати запит без аутентифікації і зафіксувати результат.
3. Реалізувати запит з використанням логіна і пароля або токена.
4. Проаналізувати відповідь сервера.
5. Дослідити механізм передачі даних аутентифікації.
6. Виконати запит до ресурсу з обмеженим доступом.
7. Проаналізувати відмінності між успішним і неуспішним доступом.
8. Дослідити поведінку сервісу при некоректних даних.
9. Зафіксувати результати виконаних запитів.
10. Сформулювати висновки.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно обрати вебсервіс або API, який підтримує механізми аутентифікації та авторизації. Це може бути відкритий сервіс із тестовим доступом або локально розгорнутий застосунок. Необхідно отримати базову інформацію про спосіб аутентифікації, що використовується у сервісі, зокрема тип облікових даних або формат токена.

На першому етапі необхідно виконати HTTP-запит до обраного ресурсу без передачі даних аутентифікації. Потрібно зафіксувати результат виконання запиту, зокрема код статусу відповіді, текст повідомлення та заголовки. Необхідно визначити, чи дозволяється доступ до ресурсу без аутентифікації, і у разі відмови зафіксувати відповідне повідомлення сервера.

На наступному етапі необхідно реалізувати запит із використанням облікових даних або токена доступу. Для цього потрібно сформувати запит із додаванням необхідних параметрів аутентифікації. Дані можуть передаватися у заголовках або у тілі запиту залежно від реалізації сервісу. Після виконання запиту необхідно отримати відповідь і зафіксувати її параметри.

Далі необхідно проаналізувати структуру відповіді сервера. Потрібно визначити, чи відрізняється відповідь від попереднього запиту без аутентифікації, а також які дані повертаються у разі успішного доступу. Особливу увагу слід приділити кодам статусу і наявності службових повідомлень.

На наступному етапі необхідно виконати запити до різних ресурсів сервісу, використовуючи ті самі облікові дані. Потрібно перевірити, які ресурси доступні, а до яких доступ обмежений. Це дозволяє дослідити механізм авторизації та визначити рівень прав доступу.

Особливу увагу необхідно приділити обробці помилок. Потрібно виконати запити з некоректними обліковими даними, зміненим або відсутнім токеном. Необхідно зафіксувати реакцію сервісу, зокрема коди статусу та повідомлення про помилки. Це дозволяє оцінити, як система реагує на порушення правил доступу.

У процесі виконання роботи необхідно зафіксувати всі результати. Для кожного запиту потрібно записати параметри запиту, код статусу, отримані дані та коментар щодо результату. Це дозволяє виконати подальший аналіз і узагальнення.

Додатково доцільно перевірити вплив зміни параметрів запиту, наприклад зміну заголовків або формату передавання даних. Це дозволяє краще зрозуміти особливості реалізації механізмів аутентифікації.

На завершальному етапі необхідно виконати аналіз отриманих результатів. Потрібно оцінити ефективність механізмів аутентифікації та авторизації, визначити їх сильні і слабкі сторони, а також сформулювати висновки щодо рівня захисту досліджуваного сервісу.

Контрольні питання

1. Що таке аутентифікація?
2. Що таке авторизація?
3. У чому різниця між аутентифікацією і авторизацією?
4. Які методи аутентифікації існують?
5. Що таке токен доступу?
6. Як передаються дані аутентифікації у HTTP-запитах?
7. Що таке контроль доступу?
8. Як визначаються права користувача?
9. Які загрози існують для механізмів аутентифікації?
10. Як забезпечується безпека передавання даних?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис використаних утиліт;
– результати виконання команд;

– аналіз отриманих даних;
– висновки.

Практична робота 13. Дослідження методів захисту даних під час передачі у мережі

Мета роботи – дослідити методи захисту даних під час передачі у мережі та принципи використання криптографічних механізмів.

Ключові положення

Захист даних під час передачі у мережі є одним із ключових аспектів безпеки інфокомунікаційних систем. У процесі передавання інформація проходить через різні мережеві вузли, маршрутизатори та канали зв'язку, що створює потенційні ризики її перехоплення, модифікації або підміни. У відкритих мережах ці загрози є особливо актуальними, оскільки дані можуть бути доступні для сторонніх осіб. Для запобігання таким ситуаціям використовуються криптографічні механізми, які забезпечують конфіденційність, цілісність та автентичність переданої інформації.

Конфіденційність означає, що дані доступні лише авторизованим сторонам. Для її забезпечення використовується шифрування, яке полягає у перетворенні відкритих даних у зашифрований вигляд за допомогою криптографічного алгоритму і ключа. Без наявності відповідного ключа розшифрування даних є неможливим або надзвичайно складним. У результаті навіть у разі перехоплення даних вони залишаються недоступними для зловмисника.

У сучасних мережах для захисту передавання даних широко використовується протокол TLS. Він забезпечує встановлення захищеного каналу зв'язку між клієнтом і сервером. Під час встановлення з'єднання відбувається обмін службовою інформацією, перевірка автентичності сторін і узгодження параметрів шифрування. Після цього весь трафік передається у зашифрованому вигляді, що гарантує конфіденційність даних.

Цілісність даних означає, що інформація не була змінена під час передачі. Для її забезпечення використовуються хеш-функції, які формують контрольне значення на основі вмісту повідомлення. Навіть незначна зміна даних призводить до зміни хешу, що дозволяє виявити порушення. У мережевих протоколах хеш-функції часто використовуються разом із механізмами шифрування, що забезпечує комплексний захист.

Автентичність даних передбачає підтвердження джерела інформації. Для цього використовуються цифрові сертифікати та електронні підписи. Сертифікат містить інформацію про власника і підтверджується довіреним центром сертифікації. Під час встановлення захищеного з'єднання клієнт перевіряє сертифікат сервера, що дозволяє переконатися у його справжності. Це є важливим для запобігання атакам типу «людина посередині».

У практиці використовується поєднання симетричних і асиметричних алгоритмів шифрування. Асиметричні алгоритми застосовуються для безпечного обміну ключами між сторонами. Після встановлення спільного ключа використовується симетричне шифрування, яке є більш швидким і ефективним для передавання великих обсягів даних. Такий підхід дозволяє поєднати високий рівень безпеки і продуктивності.

Важливим аспектом є правильна організація процесу встановлення захищеного з'єднання. Необхідно перевіряти сертифікати, контролювати термін їх дії і використовувати сучасні алгоритми шифрування. Використання застарілих або небезпечних алгоритмів може призвести до зниження рівня захисту.

Окрім технічних засобів, важливу роль відіграє правильна конфігурація системи. Наприклад, необхідно забороняти використання незахищених протоколів, таких як HTTP, для передавання конфіденційних даних. Натомість слід використовувати HTTPS, який є захищеною версією HTTP і базується на використанні TLS. Це дозволяє забезпечити безпечну взаємодію у вебзастосунках.

У процесі передавання даних можуть виникати різні загрози, зокрема перехоплення трафіку, підміна даних, повторне відтворення повідомлень або спроби несанкціонованого доступу. Використання криптографічних механізмів дозволяє ефективно протидіяти цим загрозам і забезпечити захист інформації.

У сучасних інфокомунікаційних системах захист даних під час передачі є обов'язковою вимогою. Він застосовується у вебсервісах, мобільних застосунках, системах електронної комерції та інших сферах. Наявність захищеного каналу зв'язку є критично важливою для забезпечення довіри користувачів і збереження конфіденційної інформації.

Таким чином, захист даних під час передачі базується на використанні криптографічних механізмів, що забезпечують конфіденційність, цілісність та автентичність інформації. Використання сучасних протоколів, правильна конфігурація систем і застосування перевірених алгоритмів дозволяють створити надійне середовище для обміну даними у мережі.

Практичне завдання

1. Виконати HTTP-запит до вебсервісу без використання шифрування.
2. Виконати аналогічний запит через захищений протокол HTTPS.
3. Порівняти результати виконання запитів.
4. Проаналізувати сертифікат вебсервісу.
5. Визначити параметри захищеного з'єднання.
6. Дослідити використання шифрування у передачі даних.
7. Зафіксувати відмінності між захищеним і незахищеним з'єднанням.
8. Проаналізувати ризики використання незахищених протоколів.
9. Виконати декілька запитів і зафіксувати результати.
10. Сформулювати висновки.

Методичні вказівки до виконання практичного завдання

Перед початком виконання роботи необхідно обрати вебсервіс, який підтримує як незахищений доступ через HTTP, так і захищений доступ через HTTPS. Необхідно переконатися, що обраний ресурс доступний у обох режимах, що дозволяє виконати коректне порівняння. Доцільно зафіксувати адресу сервісу та підготувати інструменти для виконання HTTP-запитів.

На першому етапі необхідно виконати запит до обраного ресурсу через протокол HTTP. Для цього потрібно використати браузер, консольну утиліту або програмний

застосунок. Отриману відповідь необхідно зафіксувати, зокрема код статусу, вміст відповіді та заголовки. Особливу увагу слід звернути на те, що дані передаються у відкритому вигляді.

Далі необхідно виконати аналогічний запит через протокол HTTPS. Потрібно використати ту саму адресу ресурсу із зазначенням захищеного протоколу. Отриману відповідь також необхідно зафіксувати і порівняти з попередньою. Слід визначити, чи відрізняється структура відповіді та чи змінюються заголовки.

На наступному етапі необхідно виконати аналіз сертифіката безпеки. Для цього потрібно отримати інформацію про сертифікат сервера за допомогою браузера або спеціалізованих утиліт. Необхідно визначити, ким виданий сертифікат, термін його дії, а також основні параметри, що характеризують рівень захисту. Отримані дані слід зафіксувати.

Після цього необхідно порівняти процес передавання даних у випадку використання HTTP і HTTPS. Потрібно визначити, які дані передаються у відкритому вигляді при використанні HTTP, і які захищаються при використанні HTTPS. Необхідно звернути увагу на відмінності у структурі з'єднання і способах обміну інформацією.

У процесі виконання роботи доцільно використати консольні утиліти або програмні засоби для аналізу мережевих з'єднань. Це дозволяє отримати детальну інформацію про параметри з'єднання, процес встановлення захищеного каналу та використані алгоритми. Отримані результати необхідно зафіксувати.

Особливу увагу слід приділити обробці можливих помилок. Необхідно перевірити, як система реагує на недійсний або прострочений сертифікат, а також на відсутність захищеного з'єднання. Потрібно зафіксувати відповідні повідомлення і проаналізувати їх.

На завершальному етапі необхідно виконати аналіз отриманих результатів. Потрібно оцінити відмінності між захищеним і незахищеним передаванням даних, визначити переваги використання HTTPS та зробити висновки щодо ефективності криптографічних механізмів у забезпеченні безпеки мережевих систем.

Контрольні питання

1. Що таке захист даних під час передачі?
2. Які основні цілі криптографії?
3. Що таке шифрування?
4. У чому різниця між симетричним і асиметричним шифруванням?
5. Що таке TLS?
6. Чим відрізняється HTTP від HTTPS?
7. Що таке цифровий сертифікат?
8. Яку роль відіграють хеш-функції?
9. Які загрози існують при передачі даних?
10. Чому важливо використовувати захищені протоколи?

Зміст звіту

– назва практичної роботи;
– мета виконання;
– опис використаних утиліт;
– результати виконання команд;
– аналіз отриманих даних;
– висновки.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

1. Kurose, James F., Ross, Keith W. Computer Networking: A Top-Down Approach. 8th ed. Pearson, 2021. 797 p. ISBN: 9781292405513, 1292405511.

2. Tanenbaum, Andrew S., Wetherall, David J. Computer Networks. 6th ed. Pearson, 2021. 944 p. ISBN: 9781292374062, 1292374063.

3. Newman, Sam. Building Microservices. 2nd ed. O'Reilly Media, 2021. 616 p. ISBN: 9781492033998, 1492033995.

4. Stallings, William. Network Security Essentials: Applications and Standards, Global Edition. Pearson, 2018. 461 p. ISBN: 9781292154916, 1292154918.

5. Hanes, D., Salgueiro, G., Grossetete, P., Barton, R., Henry, J. IoT Fundamentals: Networking Technologies, Protocols, and Use Cases for the Internet of Things. Pearson Education. 2017. 576 p. ISBN: 9780134307084, 0134307089

Допоміжна

1. Burns, Brendan, Beda, Joe, Hightower, Kelsey. Kubernetes: Up and Running. 2nd ed. O'Reilly Media, 2019. 255 p. ISBN: 9781492046523, 1492046523.

2. Richardson, Leonard, Amundsen, Mike. RESTful Web APIs. O'Reilly Media, 2013. 408 p. ISBN: 9781449359737, 1449359736.

3. Banks, Andrew, Gupta, Rahul. MQTT Version 3.1.1. OASIS Standard. OASIS, 2014.

4. Ruckemann, C. (2013). Integrated Information and Computing Systems for Natural, Spatial, and Social Sciences. Information Science Reference. 2013. 543 p. ISBN: 9781466621916, 1466621915

5. Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 19th International Conference, NEW2AN 2019, and 12th Conference, RuSMART 2019, St. Petersburg, Russia, August 26–28, 2019, Proceedings. Springer International Publishing. 2019. 759 p. ISBN: 9783030308599, 3030308596

Інформаційні ресурси

1. RFC Editor. Request for Comments Database. https://www.rfc-editor.org
2. Internet Engineering Task Force (IETF). https://www.ietf.org
3. Open Web Application Security Project (OWASP). https://owasp.org
4. Mozilla Developer Network Web Docs (HTTP, REST API). https://developer.mozilla.org
5. OASIS Open Standards (MQTT, AMQP). https://www.oasis-open.org

Навчальне видання

Мірошник Марина Анатоліївна

ПРОГРАМУВАННЯ ІНФОКОМУНІКАЦІЙНИХ СЕРВІСІВ І СИСТЕМ

Методичні вказівки до практичних робіт для здобувачів вищої освіти,
які навчаються за спеціальностями галузі «Інформаційні технології»

Українською мовою