МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

# МЕТОДИЧНІ ВКАЗІВКИ

## до лабораторних робіт з курсу «Комп'ютерні мережі»

## Частина 2

для студентів спеціальності
122 – Комп'ютерні науки

Затверджено
редакційно–видавничою
радою університету,
протокол № 3 від 6.11.2019 р.

Харків
НТУ ХПІ
2019

Методичні вказівки до лабораторної робот з курсу «Комп’ютерні мережі» для студентів спеціальності 122 – Комп’ютерні науки, / уклад. Ю. С. Шахновський. – Харків : НТУ «ХПІ», 2019. – 26 с.

Укладач Ю. С. Шахновський

Рецензент Л. М. Любчик

Кафедра системного аналізу та інформаційно–аналітичних технологій

## Вступ

Друга частина курсу присвячена вивченню найбільш застосовуваних у мережі Internet протоколів користувача. Перша лабораторна робота містіть вивчення засобів встановлення зв’язку у мережі за допомогою моделі клієнт – сервер. Друга робота присвячена знайомству с роботою електронної пошти. Третя робота вивчає працю с WEB клієнтом та налаштування зв’язку з прокси – сервером. Остання, четверта робота присвячена розробці програм, які обмінюються даними через мережу. Для цього застосовується бібліотека сокетів. Методичні вказівки містять 4 лабораторні роботи.

## Лабораторна робота 1

## МОДЕЛЬ КЛІЄНТ – СЕРВЕР

Мета роботи : навчитися працювати з програмами, що реалізують протоколи telnet та ftp.

### *1.1. Загальні відомості*

Для обміну даними по мережі повинен виконуватися обмін даними між двома програмами. При цьому на початку обміну ці програми не є рівноправними. Одна програма, яка називається сервером, надає послугу, а друга, яка запитує послугу і отримує її, називається клієнтом. Програми клієнт і сервер можуть перебувати як на одному комп’ютері, так і на різних комп’ютерах, зв’язаних мережею. Мережа, яка реалізує засоби, при цьому є середовищем, через яке ці програми взаємодіють між собою. Комп’ютери, на яких працюють ці програми, також можна назвати клієнтом і сервером. Але первинна при цьому роль програм.

Програма клієнт, виконання якої почалося у вашій системі, коли запущений запит на зв’язок по мережі, повинна:

1. Установити мережне з’єднання з сервером за допомогою протоколу TCP.

2. Прийняти від користувача вхідні дані в зручній для нього формі.

3. Перетворити ці вхідні дані в стандартну форму для передачі по мережі і послати їх серверу.

4. Прийняти від сервера вхідні дані в мережному форматі.

5. Переформатувати отримані дані для відображення на екрані вашого терміналу.

Мета роботи: навчитися передавати дані по мережі між програмами на різних комп’ютерах за допомогою стандарту сокетів.

Програма сервер виконується на комп’ютері, що надає послугу. Якщо програма сервер не працює, то запитана клієнтом послуга не доступна. Послуги надають спеціальні програми, які у WINDOWS називають резидентними, а в UNIX – демонами. Коли до них немає звернення, вони знаходяться в пам’яті і нічого не роблять. При зверненні вони виконують покладені на них функції.

Якщо програма сервер готова приймати запити, то вона виконує такі дії:

1. Інформує мережне програмне забезпечення про те, що вона готова до встановлення з’єднувань.

2. Чекає запит в стандартній формі.

3. Обслуговує цей запит.

4. Посилає результати назад програмі клієнту у стандартному форматі.

5. Очікує наступний запит.

В Internet взаємодіють сервер і клієнт, що виконують різні функції. Однією з функцій, зручних при використанні мереж, є робота на віддаленому комп’ютері. У цьому випадку користувач, сидячи за екраном і клавіатурою

одного комп’ютера, використовує процесор, пам’ять, зовнішні пристрої (відмінні від тих, які застосовуються як інтерфейс) іншого комп’ютера. Таку взаємодію сервера і клієнта називають віддаленим доступом. Віддалений доступ до іншого комп’ютера може містити або не містити графічні елементи. Але програми, що дозволяють графічний віддалений доступ, не стандартизовані щодо обладнання та операційної системи. Для передачі даних у текстовій формі є програми, загальні для всіх операційних систем і будь-яких пристроїв вводу і виводу. Протокол, за яким ці програми працюють, називається telnet. Назва telnet використовується і для найпоширенішої версії програми клієнта, яка виконує функції неграфічного віддаленого доступу. Клієнт telnet – це проста програма з текстовим інтерфейсом, що дозволяє підключатися до іншого комп’ютера через Інтернет. Якщо власник або адміністратор іншого комп’ютера дав вам право підключатися до нього, програма telnet дозволить вам вводити команди для доступу до програм і служб, які знаходяться на віддаленому комп’ютері, як ніби ви працюєте безпосередньо за ним. Клієнт telnet можна використовувати для різних завдань, у тому числі для доступу до електронної пошти, баз даних або файлів. Стандартний клієнт telnet доступний для використання з WINDOWS або UNIX простим викликом програми з командного рядка. Крім нього, використовуються інші клієнтські програми, з більш зручним інтерфейсом. Наприклад, програма putty.

Можливий варіант, коли на одному комп’ютері працюють кілька програм серверів. У цьому випадку потрібно вміти визначати, якому з серверів віддати пакет, який прийшов на комп’ютер. Для раз в’язання цієї проблеми використовують поняття «порт». Це не технічний пристрій, а слово (два байти) в пакеті. Ці два байти служать селектором програми, якій пакет призначений. Пакет буде відісланий тій програмі сервера, яка відповідає за порт, заданий в пакеті. Порти, за які відповідають сервери, не перетинаються. Для установлення зв’язку між клієнтом і сервером вони повинні використовувати загальний порт. Це досягається за рахунок узгодження між

програмістами, які пишуть програми клієнт і сервер. Вони вибирають один і той же порт. Але необхідно, щоб цей порт збігався з портами інших програм. У цілому це питання не стандартизоване, і бувають накладки. Але за найчастіше використовуваними програмами закріплені стандартні порти. Так, програми протоколу telnet за замовчуванням використовують порт 23. Список портів за замовчуванням можна знайти в UNIX у файлі /etc/services. А в WINDOWS у файлі c:\windows\system32\drivers\etc\services. Програми протоколу telnet можна використовувати для звернення до портів, відмінних від стандартного порту telnet. Якщо відомі команди протоколу, до якого звертаються за допомогою telnet, то з цим протоколом можна працювати, використовуючи клієнт telnet. Хоча й у менш зручній формі, ніж використовуючи клієнти цих протоколів. Ще одна можливість, що виникає при використанні telnet з нестандартними портами, це можливість перевірки, які сервери активні на комп'ютері, з яким намагаємося встановити зв'язок.

Інша важлива функція – мереж це передача файлів між комп'ютерами. Для цієї функції використовується стандартний протокол FTP (File Transfer Protocol). FTP дозволяє підключатися до серверів цього протоколу і переглядати вміст каталогів, завантажувати файли з сервера або на сервер. Формально це щось на зразок підключення до якоїсь папки, яка знаходиться на іншому комп'ютері/сервері, використовуючи мережу або Інтернет. У випадку, якщо передача файлу була перервана з яких-небудь причин, протокол передбачає продовження підкачки файлу, що буває дуже зручно при передачі великих файлів.

Для роботи з протоколом FTP можна використовувати стандартний клієнт командного рядка. Він викликається командою ftp під різними операційними системами. Для зручності користувачів існує багато клієнтських над налаштувань над ftp. Їх загальний принцип полягає в тому, що маніпуляції з файлами, що знаходяться на різних комп'ютерах, з'єднаних мережею, відбуваються так само, як маніпуляції за допомогою звичайних файл–менеджерів з файлами одного комп'ютера. Дії користувача в такій

оболонці перетворяться в стандартні команди ftp перед передачею в мережу. При роботі з FTP потрібно пам’ятати, що в передачі даних використовуються два комп’ютери: локальний, на якому запущений клієнт, і віддалений, на якому працює сервер.

### *1.2. Завдання на роботу*

1. У провіднику вибрати пункт меню «Мережа». У ньому знайти папку public\utilites\development\putty. запустити програму putty.exe
2. Встановити зв’язок з комп’ютером 172.16.201.1, використовуючи логін stud1, stud2 ... stud6. З паролем stud. Цей комп’ютер працює під ОС UNIX.
3. Виконати кілька команд на віддаленому комп’ютері. Наприклад ls, ps, ls -a. Переконатися, що команди виконуються, використовуючи пам’ять і процесор віддаленої машини. А локальна машина служить тільки для введення команд і відображення результатів на клавіатурі.
4. Навчитися вести протокол сеансу роботи. У telnet ця функція називається лог. Для цього вибрати в клієнті putty лівий верхній кут мишкою. У меню вибрати пункт «Change Settings», а в ньому пункт «Logging». Встановити запис всіх дій у файл і вибрати шлях до файлу в папці, в якій у вас є права доступу. Подати декілька команд. Після цього закрити сеанс зв’язку з віддаленим комп’ютером. Знайти файл, який вийшов на локальному комп’ютері.
5. Створити новий сеанс зв’язку. На віддаленому комп’ютері запустити програму telnet і з її допомогою увійти на віддалений комп’ютер вдруге. Тепер у нас відкриті два сеанси зв’язку. Перший сеанс з’єднує наш локальний комп’ютер (А) з віддаленим комп’ютером (В). Другий сеанс з’єднує віддалений комп’ютер В з іншим віддаленим комп’ютером С. При цьому в другому сеансі В виступає як клієнт, а С як сервер. Для зручності експерименту у нас як В і С використовується один комп’ютер. Але при цьому відкрито два сеанси зв’язку. Програма telnet запущена на В має

текстовий інтерфейс. В основному режимі її робота відповідає протоколу telnet і збігається з putty. Якщо нам потрібно змінити налаштування програми, то перехід в керуючий режим здійснюється натисканням на клавіатурі спеціального символа, званого escape-символом. За умовчанням як escape символ використовується «Ctrl+]». Перейшовши в керуючий режим, можна набрати символ «?» і отримати підказку, які команди доступні в цьому режимі. Більш детальну підказку можна отримати, вводячи ім'я команди і знак питання після неї. Знайдіть, як у цьому клієнті виконати створення лога.

6. Якщо маємо кілька транзитивних сеансів з використанням програми telnet, то необхідно вміти управляти кожним з цих сеансів. Для цього використовується заміна escape символів на інші, відмінні від стандартного символу. Якщо escape символ у кожному сеансі зв'язку унікальний, то проблем з вибором сеансу, якому призначена команда управління, не виникає. У telnet для зміни escape символа потрібно перейти в керуючий режим за допомогою старого escape символа, а потім подати команду «set escape Ctrl+[». Можна використовувати й інші символи. Але вибирати escape символ потрібно так, щоб він не використовувався в сеансі зв'язку і не заважав звичайній роботі в клієнті.

7. Закрити всі сеанси зв'язку. Запустити putty. Крім можливості задання віддаленого комп'ютера, клієнти telnet також дають можливість поставити порт, за яким буде встановлено зв'язок. Для віддаленого доступу за замовчуванням використовується порт 23. Вам потрібно навчитися користуватися портами, відмінними від стандартного. В клієнті putty для цього використовується поле введення поряд з полем вводу IP-адреси. В клієнті telnet поле порту вводиться після IP адреси через пробіл. Встановлюючи сеанси зв'язку з різними серверами кафедри з використанням різних портів, потрібно навчитися визначати, на яких машинах запущені серверні програми.

Перейдемо до вивчення ftp. Ftp сервер встановлений за адресою 172.16.201.200 і для зв’язку з ним можна використовувати ім’я і пароль, які збігаються з тими, що ви використовували для telnet.

1. Встановити сеанс з’єднання з сервером з командного рядка ftp 172.16.201.200.

2. За допомогою команди help подивитися список доступних команд.

3. За допомогою команд get і put навчитися переміщати файли між локальним і віддаленим комп’ютерами.

4. Навчитися переміщати, передавати по мережі виконавчі файли, для цього передати здійсненний файл у двійковому і текстовому режимі. Запуском файлу перевірити, який режим необхідний для передачі двійкових файлів.

5. Використовуючи команди mput і mget переслати всі файли з директорії. Навчитися користуватися командою prompt.

6. Закрити сеанс ftp за допомогою команди quit.

7. УВстановити зв’язок з сервером ftp за адресою 172.16.20.5 з використанням анонімного ftp.

### *Контрольні запитання*

1. Яка з програм сервер або клієнт починає з’єднання?

2. Назвіть кілька стандартних портів програм, відмінних від telnet.

3. Для чого застосовується протокол сеансу роботи?

4. Що таке транзитивне віддалене підключення?

5. Для чого використовується escape символ?

6. Який режим необхідний для передачі по мережі файлів, які містять рисунки?

7. У чому різниця між командами роботи з файловою системою cd і lcd?

8. Поясніть різницю роботи між командами put і get.

9. Для чого використовується анонімний ftp?

10. Які переваги у множинної пересилці файлів?

## Лабораторна робота 2

## ЕЛЕКТРОННА ПОШТА

Мета роботи : навчитися працювати з електронною поштою.

### *2.1. Загальні відомості*

Електронна пошта (e-mail, від electronic mail) – технологія і надані нею послуги з пересилання і отримання електронних повідомлень (званих «листи» або «електронні листи») розподіленою (в тому числі глобальною) комп'ютерною мережею.

Електронна пошта за складом елементів та принципом роботи практично повторює систему звичайної (паперової) пошти, запозичуючи як терміни (пошта, лист, конверт, вкладення, ящик, доставка та інші), так і характерні особливості – простоту використання, затримки передачі повідомлень, достатню надійність і в той же час відсутність гарантії доставки.

Переваги електронної пошти: легко сприймаються і запам'ятовуються людиною адреси вигляду username@domainname (наприклад, somebody@example.com); можливість передачі як простого тексту, так і тексту, який має форматну структуру, а також довільних файлів; незалежність серверів (у загальному випадку вони звертаються один до одного безпосередньо); досить висока надійність доставки повідомлення; простота використання людиною і програмами.

Недоліки електронної пошти: наявність такого явища, як спам (масові рекламні та вірусні розсилки); теоретична неможливість гарантованої доставки конкретного листа; можливі затримки доставки повідомлення (до

декількох діб); обмеження на розмір одного повідомлення і на загальний розмір повідомлень у поштовій скриньці (персональні для користувачів).

Загальноприйнятим у світі протоколом обміну електронною поштою є SMTP (англ. Simple mail transfer protocol – простий протокол передачі пошти). Він використовує DNS для визначення правил пересилання пошти. DNS дозволяє вказати як приймаючий сервер будь-який вузол Інтернету. Це може використовуватися для налаштування релеінга (пересилання) пошти через треті сервери.

### *2.2. Завдання на роботу*

1. Надіслати лист самому собі з метою перевірки функціонування пошти.

2. Встановити текстовий режим надсилання листа.

3. Установити правила маркування тексту, який містився в листі, на який буде дано відповідь.

4. Розбитися на пари. Надіслати лист своєму сусідові по парі. Лист повинен містити в собі три питання, кожне питання має бути розташоване в новому рядку. Отримавши листа від сусіда, необхідно відповісти на питання. При цьому відповідь на кожне питання розташовувати під поміченим текстом питання. Кілька разів обмінятися листами, з уточненнями відповідей з попереднього пункту. Розібратися з поняттям тред повідомлень.

5. Надіслати сусідові лист із вкладеним рисунком.

6. Надіслати лист із вкладеним виконуваним файлом. Розібратися з проблемами, які при цьому виникають.

7. Створити список розсилки з усіх студентів вашої групи. Надіслати лист за допомогою списку розсилки.

8. Надіслати лист зі звітом про виконану роботу за e-mail адресою, яка вказана викладачем.

### *Контрольні запитання*

1. У чому перевага автоматичної позначки тексту, який був в початковому листі, при відповіді на лист?

2. Які проблеми виникають при надсиланні виконуваного файлу? Способи їх розв'язання.

3. Які переваги у списку розсилки в порівнянні з переліком множини адрес, на які має бути відправлено лист?

4. З яких частин складається адреса електронної пошти?

5. У чому різниця при пересиланні текстових і двійкових повідомлень?

6. Що таке spam?

7. Чому відсилання файлів, які можуть бути виконані, створює загрозу для приймаючої сторони?

8. За рахунок чого можлива асинхронність роботи відправника і одержувача електронної пошти?

9. Передача інформації електронною поштою вважається слабо захищеною від перехоплення. Чому?

10. Порівняйте переваги й недоліки передачі даних по e-mail зі звичайною поштою. З телефонною розмовою. З передачею інформації по Skype.

## Лабораторна робота 3

## ВИКОРИСТАННЯ WEB. РОБОТА ЧЕРЕЗ PROX. ІНФОРМАЦІЙНІ ПОШУКОВІ СЕРВЕРА.

Мета роботи: вивчити можливості клієнтських програм для WWW. Навчиться налаштовувати роботу через proxy. Дослідити пошук у мережі за допомогою пошукових веб-серверів і реєструвати в мережі свій сайт.

### *3.1 Загальні положення*

Браузер, або ще так званий веб-оглядач, це спеціальна програма, яка дозволяє переглядати ті чи інші сайти. Браузер надсилає запит серверу на

отримання будь-якої інформації або даних. Отримавши відповідь, інтерпретує все це спеціальним чином і показує веб-сторінку за допомогою засобів доступних на вашому засобі відображення.

Для розробки і зберігання веб-сторінок використовується мова HTML (Hyper Text Markup Language). Ця мова належить до класу SGML мов. Відмінність SGML мов у тому, що документ складається з корисної інформації, яка укладена в спеціальні «дужки» – теги. Теги описують, як інформація буде подана на пристрої, що застосовується користувачем, для відображення документа. У кожного елемента є відкриваючі і закриваючі теги, які визначають межі дії правил. У тезі, що відкриває, на подання документа впливають ім'я тега і додаткові параметри – атрибути.

Приклад:

```
<X sss = «білий»>
текст
</ X>
```

Документи на HTML називають гіпертекстом тому, що в них можна вставляти посилання на інші документи або на інші частини поточного документа. Посилання в документ вставляють за допомогою тега <A>. При цьому існує дві версії тега. Одна що використовується щоб задати аналог мітки в тексті <A name = #«ідентифікатор»>, задає якір всередині Web сторінки. Якорем називається закладка всередині сторінки, яку можна вказати як мету посилання. При використанні посилання, яке вказує на якір, перехід відбувається не на початок сторінки, а на закладку, задану якорем. Другий варіант оператора А має вигляд <A href=«URL»>, де URL задає документ в Інтернет, до якого здійснюється перехід при виборі посилання. Перехід може бути заданий як до початку документа, так і до його частини, визначеної за допомогою першого варіанта оператора А.

URL – це єдиний указівник ресурсів (Uniform Resource Locator). Він служить стандартизованим способом запису адреси ресурсу в мережі Інтернет.

URL служить для вказівки місця розташування ресурсу в мережі. Він має таку схему:

<*схема*>: // <*логін*>: <*пароль*> @ <*хост*>: <*порт*> / <*URL-шлях*>? <*Параметри*> # <*якір*>

У цьому записі:

*схема* – схема звернення до ресурсу; в більшості випадків мають на увазі мережний протокол;

*логін* – ім’я користувача, яке використовується для доступу до ресурсу;

*пароль* – пароль зазначеного користувача;

*хост* – повністю прописане доменне ім’я хоста в системі DNS або IP-адреса хоста у формі чотирьох груп десяткових чисел, розділених точками;

*порт* – порт хоста для підключення;

*URL-шлях* – уточнююча інформація про місце знаходження ресурсу;

*параметри* – рядок запиту з переданими на сервер параметрами. Роздільник параметрів – знак &;

*якір* – описаний вище.

Проксі-сервер – служба в комп’ютерних мережах, що дозволяє клієнтам виконувати непрямі запити до інших мережних служб. Спочатку клієнт підключається до проксі-сервера і запитує який-небудь ресурс, розташований на іншому сервері. Потім проксі-сервер або підключається до вказаного сервера і отримує ресурс у нього, або повертає ресурс із власного кешу (у випадках, якщо проксі має свій кеш). У деяких випадках запит клієнта або відповідь сервера може бути змінений проксі-сервером у певних цілях. Проксі-сервер дозволяє захищати комп’ютер клієнта від деяких мережних атак і допомагає зберігати анонімність клієнта.

Проксі-сервери застосовуються для таких цілей:

- Забезпечення доступу комп’ютерів локальної мережі до мережі Інтернет
- Кешування даних: якщо часто відбуваються звернення до одних і тих же зовнішніх ресурсів, то можна тримати їх копію на проксі-сервері і видавати за запитом, знижуючи тим самим навантаження на канал у

зовнішню мережу і прискорюючи отримання клієнтом запитаної інформації.

- Захист локальної мережі від зовнішнього доступу: наприклад, можна налаштувати проксі-сервер так, що локальні комп'ютери будуть звертатися до зовнішніх ресурсів тільки через нього, а зовнішні комп'ютери не зможуть звертатися до локальних взагалі (вони «бачать» тільки проксі-сервер).
- Обмеження доступу з локальної мережі до зовнішньої. Наприклад, можна заборонити доступ до певних веб-сайтів, обмежити використання інтернету якимось локальним користувачам, встановлювати квоти на трафік або смугу пропускання, фільтрувати рекламу і віруси.
- Анонімізація доступу до різних ресурсів. Проксі-сервер може приховувати відомості про джерело запиту або користувача. В такому випадку цільової сервер бачить лише інформацію про проксі-сервер, наприклад, IP-адреса, але не має можливості визначити дійсне джерело запиту. Існують також проксі-сервери, що спотворюють, які передають неправдиву інформацію про справжнього користувача.
- Обхід обмежень доступу. Проксі-сервери популярні серед користувачів країн, де доступ до деяких ресурсів обмежений законодавчо і фільтрується.

Пошукова система – програмно-апаратний комплекс з веб-інтерфейсом, що надає можливість пошуку інформації в Інтернеті. Для пошуку інформації за допомогою пошукової системи користувач формулює запит. За запитом користувача пошукова система генерує сторінку результатів пошуку. Така пошукова видача може поєднувати різні типи файлів, наприклад: веб-сторінки, зображення, аудіофайли. Мета пошукової системи полягає в тому, щоб знаходити документи, де зазначено ключові слова, або слова будь-яким чином пов'язані з ключовими словами. Пошукова

система тим краща, чим більше документів, яки релевантні запиту користувача, вона буде повертати.

Пошукові системи працюють, зберігаючи інформацію, яку вони отримують з HTML сторінок. Пошуковий робот – програма, яка автоматично проходить по всіх посиланнях, знайдених на сторінці, і виділяє їх. Робот, ґрунтуючись на посиланнях або виходячи із заздалегідь заданого списку адрес, здійснює пошук нових документів, ще не відомих пошуковій системі.

Пошукова система аналізує вміст кожної сторінки для подальшого індексування. Слова можуть бути вилучені із заголовків, тексту сторінки або спеціальних полів – метатегов. Дані про веб-сторінки зберігаються в індексній базі даних для використання в повторних запитах. Індекс дозволяє швидко знаходити інформацію за запитом користувача. Пошукова система працює з вихідними файлами, отриманими від індексатора. Пошукова система приймає запити користувачів, обробляє їх за допомогою індексу і повертає результати пошуку.

Коли користувач уводить запит у пошукову систему (зазвичай за допомогою ключових слів), система перевіряє свій індекс і видає список найбільш підходящих веб-сторінок (відсортованого за якогось критерію), зазвичай з короткою анотацією, що містить заголовок документа і іноді частини тексту. Пошуковий індекс будується за спеціальною методикою на основі інформації, витягнутої з веб-сторінок. Більшість пошукових систем підтримує використання в запитах булевих операторів І, АБО, НЕ, що дозволяє уточнити або розширити список шуканих ключових слів. При цьому система буде шукати слова чи фрази точно так, як було введено. У деяких пошукових системах є можливість наближеного пошуку, в цьому випадку користувачі розширюють область пошуку, вказуючи відстань до ключових слів.

Корисність пошукової системи залежить від релевантності знайдених нею сторінок. Хоч мільйони веб-сторінок і можуть включати якесь слово або фразу, але одні з них можуть бути більш релевантні, популярні або

авторитетні, ніж інші. Більшість пошукових систем використовує методи ранжирування, щоб вивести в початок списку «кращі» результати. Пошукові системи вирішують, які сторінки більш релевантні і в якому порядку повинні бути показані результати, по-різному. Методи пошуку, як і сам Інтернет, з часом змінюються.

### *3.2. Завдання на роботу*

1. Запустити Internet Explorer. Завантажити будь-яку сторінку. На сторінці натиснути правою кнопкою миші і вибрати пункт «Перегляд вихідного коду». Подивитися приклад коду на HTML

2. Створити просту сторінку HTML з парою рядків тексту і посиланням на іншу, існуючу сторінку. Переконатися, що посилання працює.

3. Ознайомитися з вмістом пункту «Налаштування» в настройках. Подивитися закладки і розібратися, яка з закладок для чого використовується.

4. Знайти в опціях місце, в якому налаштовується доступ до proxy. Перевірити, що як proxy сервера вказано 172.17.10.2 і як порт – 3128. Ознайомитися з більш докладними налаштуваннями proxy в браузері. Навчитися задавати сайти, для доступу до яких proxy не використовується.

5. Зайти на пошуковий сайт. Наприклад, на google.com. Сформувати запит, який містить два слова. Запам’ятати кількість відповідей, які відповідають цьому запиту. Сформувати запит, який містить ці ж два слова в подвійних лапках. Порівняти кількість відповідей, які знайдені цим запитом з першим запитом. Пояснити результат.

6. Розібратися, як на використовуваному вами пошукові сервері отримати доступ до розширеної мови запитів (advanced search). Які додаткові можливості в порівнянні зі звичайними запитами дає використання розширеного пошуку.

7. Навчитися реєструвати сайти, які створені вами, на пошукових серверах.

### *Контрольні запитання*

1. Для яких функцій використовується web-browser?

2. У чому відмінність мови розмітки гіпертексту від звичайних мов програмування?

3. У чому відмінність гіпертексту від звичайного тексту?

4. Які переваги дає вихід в Internet через proxy? До яких обмежень приводить ця технологія?

5. Які Internet протоколи можуть працювати з використанням proxy, а які ні?

6. У чому відмінність мов простих запитів до пошукових серверів від мов розширених запитів? Чим ця різниця викликана?

7. Звідки пошукові системи одержують інформацію про нові сторінки і нові сайти, що розміщуються в Internet?

8. Чим визначається порядок, в якому розміщуються посилання на сторінки, знайдені при пошуковому запиті?

9. Для чого в URL використовується схема доступу? Який зв’язок між схемою доступу і іменами прикладних протоколів?

10. Як працюють пошукові роботи. Які заходи застосовуються, щоб пошукові роботи менше завантажували мережу?

## Лабораторна робота 4

## РОЗРОБКА ПРОСТИХ СЕРВЕРА І КЛІЄНТА

Мета роботи: навчитися передавати дані по мережі між програмами на різних комп’ютерах за допомогою стандарту сокетів.

### *4.1. Загальні положення*

Для передачі даних по мережі між двома машинами використовується технологія клієнт - сервер. Функціональна відмінність між клієнтом і сервером виявляється на етапі установки зв’язку між машинами і потім у

процесі обміну пакетами з даними відмінність відсутня. На рис 8.1 показана схема відмінностей у функціях, які виконують програми сервера і клієнта.

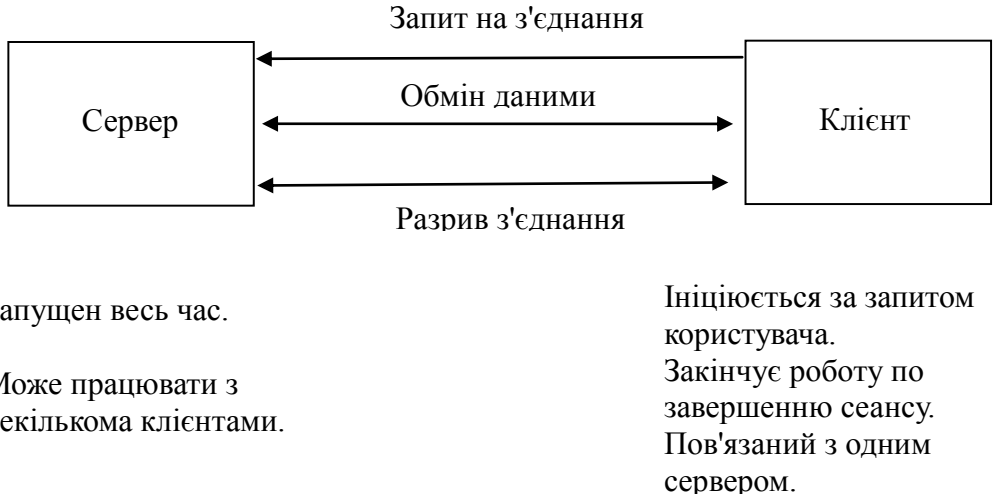


Рисунок 4.1 –. Схема взаємодії сервера і клієнта.

Через різницю виконуваних функцій алгоритми сервера і клієнта розрізняються між собою на етапі встановлення з’єднання. І програмуються по–різному. Оскільки алгоритм роботи клієнта більш простій, почнемо з нього. На рис. 4.2 показана блок–схема виклику функцій з клієнта.

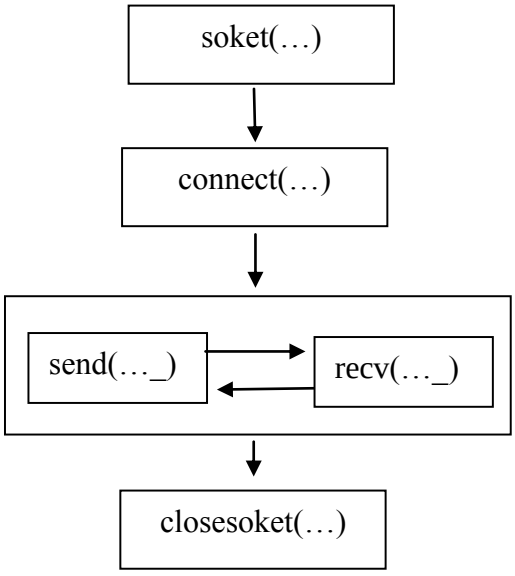


Рисунок 4.2 – Блок схема праці клієнта

Функція socket призначена для виділення місця під структури даних, в яких буде зберігатися інформація про сокеті. Її прототип:

int socket(int тип з’єднання з мережею, int тип передачі пакетів, int prt);

Функція повертає номер відкритого сокета, який буде в подальшому використовуватися у всіх функціях, які працюють з цим сокетом. Можна

провести аналогію з номером файлового дескриптора, яку повертає функція open. Перший параметр задає використовуваний протокол передачі пакетів мережного рівня. Використання протоколу IP задається константою PF_INET. Інші можливі значення цього параметра в даний час, коли Internet домінує, не використовуються. Другий параметр задає використовуваний протокол транспортного рівня. Якщо використовуємо протокол TCP, то потрібно задати SOCK_STREAM. Якщо використовуємо UDP, то задаємо SOCK_DGRAM. Третій параметр задається відмінним від 0 тільки в тому випадку, якщо використовуються нестандартні пакети, задані в другому параметрі значенням SOCK_RAW. Для TCP і UDP третій параметр завжди 0. У результаті маємо виклик:

int sd=socket(PF_INET, SOCK_DGRAM,0);

Після виділення пам’яті для сокета клієнту потрібно підключитися до сервера. Це виконує функція connect з прототипом:

int connect(int sd, struct sockaddr addres, int addr_len); де sd номер сокета, який повернула функція socket, другий параметр – це структура, яка містить дані про сервер, третій параметр це кількість байт, займаних другим параметром. Перед викликом функції необхідно заповнити три поля в структурі другого параметра. Ці поля:

а) sin_family, яке відповідає за тип використовуваної адреси і для протоколу IP має значення AF_INET;

б) sin_addr, яке містить IP-адресу сервера;

в) sin_port, яке задає порт з’єднання.

Заповнення цих полів показано в прикладі програми, яка приведена нижче.

Після того як з’єднання з сервером встановлено, обмін даними здійснюється за допомогою функцій send() і recv(), порядок виклику яких визначається протоколом, який використовує програма.

Прототип функції send:

int send(int sd, char *buf, int len, unsigned int flags).

Функція повертає кількість успішно відправлених байтів. При нормальній роботі це число збігається з len. Перший параметр це використовуваний в пересиланні сокет. Другий параметр – адреса початку буфера, який містить дані, що пересилаються. Третій параметр – довжина даних, що пересилаються, у байтах. Четвертій параметр функції дозволяє змінювати роботу сокета, який використовується у функції. Якщо зміна роботи сокета не потрібна, то четвертий параметр дорівнює 0.

Прототип функції recv:

int recv(int sd, char *buf, int len, unsigned int flags).

Сенс параметрів той же, що і для send, тільки параметр len задає максимальну кількість байтів, яку програма готова прийняти. А функція повертає кількість байтів, які реально прочитані. Якщо при зверненні функція recv повернула 0, то це ознака того, що сокет був закритий.

Функція closesocket розриває з’єднання і звільняє пам’ять, виділену для функціонування сокета.

Для роботи з системою сокетів під ОС Windows її необхідно ініціалізувати. Ініціалізацію системи сокетів під Windows виконує функція WSAStartup, а звільнення пам’яті, яка виділена під систему сокетів, виконує функцію WSACleanup. Приклад роботи з цією функцією дивіться нижче.

Блок схема роботи простий програми сервера представлена на рис 4.3.

У порівнянні з клієнтом, тут відсутній виклик функції connect і є нові функції bind, listen і accept.

Функція bind виконує реєстрацію сервера у драйвера TCP/IP машини, на якій програма сервер запускається. Цією функцією сервер говорить драйверу, що в подальшому всі пакети, які приходять на зазначений в bind порт, необхідно віддавати цьому серверу.

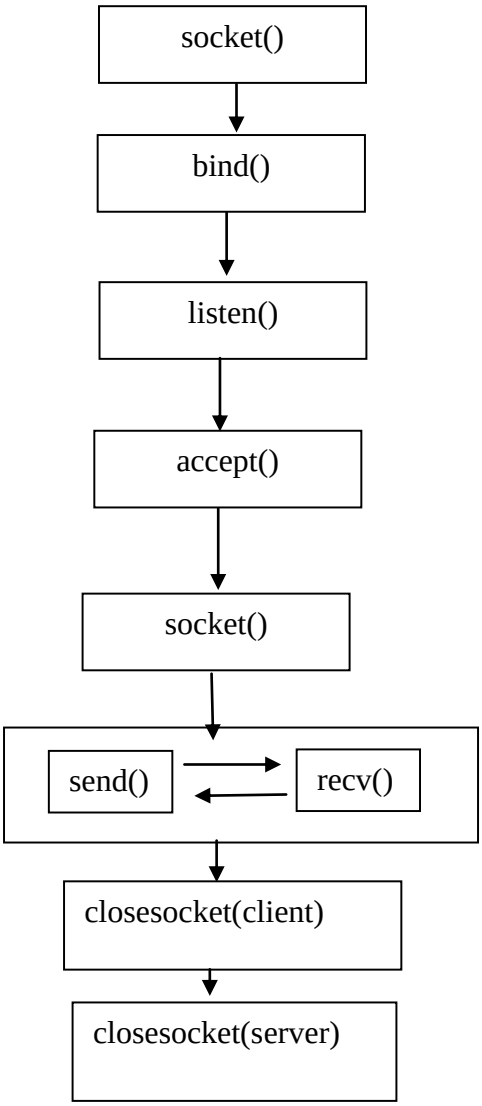


Рис 4.3 – Блок схема праці сервера

Прототип функції bind:

int bind(int sd, struct sockaddr *addr, int addr_len);

Другий параметр – це та ж структура, що і у функції connect для клієнта. Але на відміну від клієнта, в ній перед викликом bind потрібно заповнити тільки два поля - протокол і порт. А поле адреси заповнювати не треба, оскільки програма сервер і так знає адресу комп’ютера, на якому її запускають. Функція bind повертає код помилки. Якщо помилки не було, то повертає 0. Якщо помилка була, то код повернення функції буде відмінний від нуля, а значення помилки можна подивитися в системній зміної errno, якщо програм працює під UNIX і за допомогою функції WSAGetLastError, якщо програма працює під Windows.

Функція listen призначена для перекладу сокета, відкритого функцією socket в режим прослуховування. В цьому режимі сокет, який вказаний у формальної процедури не може бути використаний для читання або запису даних за допомогою функцій send і recv. А може бути використаний для отримання сигналів від нових клієнтів, які намагаються зв’язатися з сервером. Прототип функції listen:

int listen(int sd, int numslots), де параметр numslots показує, скільки пам’яті виділяти для клієнтів, які очікують у черзі на обслуговування. Це число рекомендується ставити від 3 до 5.

Прийом запитів на обслуговування клієнтів виконує функція accept з прототипом:

int accept(int sd, struct sockaddr *addr, int *addr_len).

Параметри цієї функції схожі на параметри функції connect, але на відміну від неї, функція accept отримує адресу і порт машини клієнта, яка встановлює з нею зв’язок. Тому структуру sockaddr перед викликом функції заповнювати не потрібно. І третій параметр передається за посиланням, оскільки він теж є параметром, що повертається. Значення, яке функція accept повертає – це номер ще одного сокета, що відкривається для зв’язку з клієнтом. Цей сокет буде працювати в режимі читання і запису, і він відрізняється від сокета, відкритого на прослуховування. Після закінчення роботи з поточним клієнтом необхідно закрити цей сокет і знову слухати початковий сокет.

Приклад коду програми клієнта

```
#include "stdafx.h"
#include <winsock2.h>
const int BUFMAXLEN = 80;
#define WORK_PORT 10000

int main()
{
	WSAData w;
	char straddr[] = "127.0.0.1";
	struct hostent *remoteHost;
```

```
      WSAStartup(2, &w);
      int sd = socket(PF_INET, SOCK_STREAM, 0);
      sockaddr_in Service;
      Service.sin_family = AF_INET;
      Service.sin_port = htons(WORK_PORT);
      if ((Service.sin_addr.s_addr = inet_addr(straddr)) == INADDR_NONE)
      {
            remoteHost = gethostbyname(straddr);
            Service.sin_addr.s_addr =
                        *(u_long *) remoteHost->h_addr_list[0];
      }
      connect(sd,(struct sockaddr *)& Service, sizeof(Service ));
      char buf[BUFMAXLEN];
      scanf("%s", buf);
      send(sd,buf,BUFMAXLEN,0);
      closesocket(sd);
      WSACleanup();
    return 0;
}
```

## Приклад коду програми сервера

```
#include "stdafx.h"

#pragma comment(lib, "Wsock32.lib")
#include <winsock2.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

const int BUFMAXLEN = 80;
#define WORK_PORT 1000

int main()
{
      WSAData w;

      WSAStartup(2, &w);
      int sd = socket(PF_INET, SOCK_STREAM, 0);

      sockaddr_in serv;
      int len;
      serv.sin_family = AF_INET;
```

```
        serv.sin_port = htons(WORK_PORT);
        if (bind(sd, (struct sockaddr *) &serv, sizeof(serv)))
        {
              int error = WSAGetLastError();
              printf("Error code %d",error);
              char c=getc(stdin);
              exit(1);
        }
        listen(sd, 3);
        int client = accept(sd, (struct sockaddr *)&serv, &len);
        char buf[BUFMAXLEN];
        int rc = recv(client, buf, BUFMAXLEN, 0);
        buf[rc] = 0;
        printf("%s\n", buf);
        closesocket(client);
        closesocket(sd);
        WSACleanup();
        return 0;
}
```

### *4.2 Завдання на роботу*

1. Ознайомитися до кодом програми, яку дав викладач.

2. Відкомпілювати і запустити її. Переконатися в працездатності.

3. Змінити програму, щоб вона виконувала одну з таких дій:

3.1. Клієнт звертається до порту daytime (13) сервера з адресою 172.16.201.1, отримує у відповідь від сервера рядок, яка містить поточну час і дату і видавала їх на екран.

3.2. Клієнт передає серверу ім’я файлу, у відповідь сервер посилає клієнтові вміст файлу. Це вміст записується на диск.

3.3. Програми обмінюються повідомленнями по черзі. До тих пір, поки на одній зі сторін не буде введено ключове слово «exit».

3.4. Клієнт перебирає порти на заданій машині в заданому діапазоні та виводить на екран ті з портів, на яких виявлені активні програмні сервера.

### *Контрольні запитання*

1. Чому програми сервера і клієнта мають відмінності?

2. Яка програма починає установку зв’язку при передачі даних між двома машинами в мережі?

3. Що таке мережевий порядок байтів? Як цей термін впливає на програму?

4. У чому різниця між сокетом, що задіяний на прослуховування і сокетом, що задіяний на введення та вивід?

5. Чим визначається порядок викликів функцій перекладу рядка з IP адресою і перекладу рядка з доменним іменем в двійковий код?

## СПИСОК ЛИТЕРАТУРИ

1. Антонов В. М. Сучасні комп’ютерні мережі / Валерій Миколайович Антонов. – Київ: МК-Прес, 2005. – 480 с.

2. Новиков Ю. Компьютеры, сети, Интернет: Энциклопедия: Наиболее полн. и подроб. рук. / Ю. Новиков, Д. Новиков, А. Черепанов, В. Чуркин; Под общ. ред. Ю. Новикова. – 2. изд. – Москва [ и др.]: Питер, 2003 (СПб.: ГПП печ. двор им. А.М. Горького). – 831 с

Навчальне видання

# МЕТОДИЧНІ ВКАЗІВКИ
до лабораторних робіт з курсу «Комп'ютерні мережі» частина 2
для студентів 122 – Комп'ютерні науки

Укладач: Шахновський Юрій Сергійович

Відповідальний за випуск проф. О.С. Куценко
Роботу до видання рекомендував проф. М. І. Безменов
У авторській редакції

План 2019 р., поз. 295

Підписано до друку 27.11.2019 р. Формат 60×84 1/16.Папір офсетний.
Друк –ризографія. Гарнітура Таймс. Ум. друк. арк. 1,5.
Наклад 25 прим. Зам. № _____. Ціна договірна

---

Видавничий центр НТУ «ХПІ».
Свідоцтво про державну реєстрацію ДК№ 5478 від 21.08.2017 р.
61002, Харьків, вул. Кирпичова

---

Друкарня «ФОП Пісня О.В.»
Свідоцтво про державну реєстрацію ВО2 № 248750 від 13.09.2007 р.

61002, Харків, вул. Гіршмана 16а, кв. 21, тел. (057) 764–20–29