*Лекція № **11**.*
ТЕЗИ

Веб-сервер Apache.

Попередня лекція:
www, ***URL***, URI. Структура URL.
Пртокол HTTP.
HTTP працює згідно схеми ***запит-відповідь***.
HTTP ***не зберігає стану*** між запитами.
Кожен запит а також кожна відповідь складається з стартового рядка, заголовків та тіла (даних).

Отже www (World Wide Web) працює за схемою клієнт-сервер.
Клієнтом виступає зокрема ***браузер*** — програма для перегляду HTML-документів та супутнього контенту.
Найважливішою (і найскладнішою) частиною цієї схеми є веб-сервер.

Потрібно добре розуміти, ЩО відбувається на сервері, а що на клієнті.

Призначення веб-сервера полягає в перетворенні URL в ім'я файлу та у відправці цього файлу назад клієнтові. Або в перетворенні URL в ім'я програми, виконання цієї програми та у відправці виводу цієї програми клієнтові.

Решта задач — похідні від цієї.

Коли отримуємо сторінку (HTML), то браузер ініціює отримання малюнків, файлів стилів, іншого контенту.
ПОЯСНИТИ.

Поняття динамічного і статичного контенту.
Сьогоднішні сайти в багатьох випадках передбачають інтерактивність а не просто перегляд незмінної (статичної) інформації.
Починаючи від 2005 року навіть йдеться про новий (до кінця ще не визначений) термін для означення нової “сутності”, породженою розвитком www, про Web 2.0 (на противагу 1.0)
Коли говорять про Web 2.0, мають на увазі такі масові сервіси як Wikipedia, YouTube, Flickr, LiveJournal, Gmail, GoogleMaps, соціальні мережі, сайти новин і т.п.
Все це не просто сайти а повноцінні веб-служби (напр. з власним обміном даними з сервером по XML).
Окрім власне самого контенту і взаємодії людини з цим контентом, за поняттям Web 2.0 стоять також технології (такі як AJAX, rss).

Важливо однак розуміти, що всі ці важливі зміни насправді дуже мало що міняють з технічної точки зору на роботу сервера.
Всеодно сервер віддає контент (згідно запиту клієнта) по протоколу HTTP, а клієнт його отримує по цьому ж протоколу. Як клієнт буде працювати з отриманими даними — це вже справа клієнта.

В будь-якому разі ще один важливий момент стосується факту, що основне навантаження, як і 10 років тому, так і зараз лягає на сервер. Віддача численним клієнтам контенту в режимі реального часу не належить до найпростіших задач.

Вимоги до сервера

- працювати швидко, обробляти максимум запитів, використовуючи мінімум апаратних засобів
- бути багатозадачним — одночасно працювати більше ніж з одним клієнтом
- багатозадачність з точки зору управління вмістом: зміна контенту на льоту — без зупинки сервера (відпрацювання або зовнішньої програми (CGI) або модуля (напр mod_php) apache)
- мати засоби аутентифікації
- вміти працювати в режимі шифрування трафіку
- коректно обробляти помилки
- обмінюватися з клієнтом повідомленнями про стиль і мову відповідей (заголоки типу Accept: *type/subtype* , Accept-Encoding)
- вміти пропонувати різні фрмати вихідної інформації.
- Працювати як кешуючий (proxy) сервер.
- Бути надійним, стійким до збоїв (напр до **DoS Attack - (Denial of Service)**)
- Бути безпечним. Можливість проникнення на сервер зловмисника через веб має бути зведена до мінімуму.

Всі ці задачі (а також багато інших) виконує веб-сервер ***Apache***.
http://www.apache.org
Остання на даний момент версія 2.2
Apache версії 2 суттєво відрізняється від вітки 1. (багатопотоковість – головна зміна, підтримка IPv6, деяке спрощення конфігурації, суттєві зміни API, PCRE, інші зміни)

Однак apache – не єдиний веб-сервер в UNIX (і на MS Windows). Є і інші, звичайно.
IIS (Internet Information Server – на MS Windows.

Більше 50% веб-серверів на даний момент в мережі - доля apache

LAMP (Linux-Apache-Mysql-PHP) — найпоширеніша на даний момент масова хостингова платформа.

Архітектура apache

- багатозадачний (і багатопотоковий поч. від версії 2) сервер. 1 управляючий процес і решта – запускаються і “вбиваються” ним же.
ПОЯСНИТИ!
Всі дочірні процеси працюють від непривілейованого користувача (www, web etc.)
Головний – від root, якщо сервер запущено в режимі демона.

Можна ще запускати через inetd, але це доцільно робити дуже рідко.

httpd – демон apache (якщо не зкомпільовано інакше).

Apache – від слова patch – латка.
Apache отже складається з багатьох модулів-латок: ***DSO (Dynamich Shared Modules)*** модулів.

Apache може обслуговувати як кілька IP адрес на одній машині так і масу різних доменних імен (сайтів) на одній адресі.
ПОЯСНИТИ: протоколи HTTP 1.0 та 1.1 (віртуальний хостінг та NameVirtualHost): історично згідно протоколу HTTP 1.0 сервер обслуговував один сайт (віртуальний хост) на одній адресі. Сайт міг мати псевдоніми (aliases), але не могло бути кілька різних сайтів на одній IP.
З введенням HTTP 1.1 стало можливим на одній IP мати як завгодно багато сайтів, очевидно, що під різними dns іменами (Заголовок "Host:"). Аліасінг також можливий. Тобто маємо дуже гнучку схему адресації сайтів.
Не можливо на одній IP мати лише кілька https-сайтів з різними (валідними) сертифікатами.

Віддача динамічного контенту

Для відпрацювання динамічного контенту історично першою була технологія ***CGI — Common Gateway Interface***
CGI дозволяє серверу виконувати зовнішні програми, які генерують HTMLкод. Цей код веб-сервер відправляє назад клієнту.
Як правило на сервері (чи для вірт. хоста) виділяється окрема директорія, в якій дозволяється виконання cgi-програм. Ця директорія конфігурується з дотриманням відповідних норм безпеки.
CGI-програмами можуть бути як виконувані зкомпільовані файли, так і всеможливі скріпти (php, perl, python, shell).

Як альтернатива сервер може бути доповнено ***модулями (DSO) підтримки мов сценаріїв***.
Такий модуль забезпечує середовище для виконання сценарію а також надає API для доступу дозовнішніх джерел даних та даних, отриманих від клієнта.
В apache кожен такий модуль реєструється як окремий MIME-тип і визначається розширення файлів, які будуть оброблятися цим модулем.

Другий підхід є зараз суттєво поширенішим (перший домінував раніше).

Як правило серверні сценарії діляться на такі, які включаються в HTML-файли і такі, які генерують HTML документ повністю.
Приклад першого типу — ***php (mod_php), jsp — Java Server Pages, технологія SSI (Server Side Includes).***
Другого — ***програми CGI, Java Servlets.***
При складній серверній аплікації другий підхід має більше переваг з причин меншого навантаження на сервер: apache не повинен розбирати (parsing) кожен файл.