Съдържание

[1 Увод: 3](#_Toc133505807)

[2 MySQL: 3](#_Toc133505808)

[3 Инсталирайте MySQL драйвер: 3](#_Toc133505809)

[4 Python и MYSQL: 4](#_Toc133505810)

[5 Python в MYSQL: 4](#_Toc133505811)

[6 База данни MYSQL: 5](#_Toc133505812)

[7 FTP сървър: 6](#_Toc133505813)

[7.1 Определение на FTP сървър: 7](#_Toc133505814)

[7.2 За какво се използва FTP сървър? 7](#_Toc133505815)

[7.3 Обмен на файлове с голям размер: 8](#_Toc133505816)

[7.4 Подобряване на сигурността: 8](#_Toc133505817)

[7.5 Оптимизиране на работните потоци: 8](#_Toc133505818)

[7.6 Подобряване на контрола: 8](#_Toc133505819)

[7.7 Надеждно възстановяване след бедствие: 9](#_Toc133505820)

[7.8 Какви са видовете FTP? 9](#_Toc133505821)

[7.9 Активен режим: 9](#_Toc133505822)

[7.10 Пасивен режим: 10](#_Toc133505823)

[7.11 FTP срещу облачно съхранение: 10](#_Toc133505824)

[7.12 Достъпност: 10](#_Toc133505825)

[7.13 Сигурност: 11](#_Toc133505826)

[8 Библиотеки за комуникация между сървър и клиент: 11](#_Toc133505827)

[8.1 Socket: 11](#_Toc133505828)

[8.2 Twisted: 11](#_Toc133505829)

[8.3 Flask: 12](#_Toc133505830)

[8.4 Django: 12](#_Toc133505831)

[8.5 Типове сокети: 12](#_Toc133505832)

[8.6 IP адреси и портове: 12](#_Toc133505833)

[8.7 Протокол за контрол на предаването (TCP): 13](#_Toc133505834)

[8.8 Протокол за потребителски дейтаграми (UDP): 13](#_Toc133505835)

[8.9 Интернет протокол (IP): 14](#_Toc133505836)

[8.10 Протокол за прехвърляне на хипертекст (HTTP): 14](#_Toc133505837)

[9 Сървър на django: 14](#_Toc133505838)

[10 Шини. Синхронен и асинхронен обмен: 15](#_Toc133505839)

[10.1 Асинхронен обмен: 20](#_Toc133505840)

[10.2 Синхронен обмен: 22](#_Toc133505841)

[11 Въведение в XAMPP: 25](#_Toc133505842)

[11.1 Нужда от XAMPP: 26](#_Toc133505843)

[11.2 Предимства на XAMPP: 26](#_Toc133505844)

[11.3 Недостатък на XAMPP: 27](#_Toc133505845)

[11.4 Компоненти на XAMPP: 27](#_Toc133505846)

[11.5 Приложения на XAMPP: 28](#_Toc133505847)

[12 Обяснение на кода: 29](#_Toc133505848)

[13 Използвана литература: 30](#_Toc133505849)

# Увод:

Този проект се състой от 3 независими модула. Но 3-те заедно се съчетават в един общ проект , който прави сървър за архивиране на данни. Още от като си изтеглихме темите за проектите, с моите съученици започнахме да се виждаме по-често, за да направим кода си. Колкото и да ни беше трудно, не се оплаквахме защото знаехме, че Господин Бориков ще ни помогне. Моят сървър е FTP сървър. Данните се архивират на него. FTP сървъра е софтуерно приложение, което позволява споделяне и прехвърляне на файлове между компютри в мрежа, обикновено в интернет. То използва клиент-сървър архитектура, където FTP клиентът поисква и изтегля файлове от FTP сървъра.

# MySQL:

MySQL е широко използвана система за управление на релационни бази данни (RDBMS). MySQL е безплатен и с отворен код. MySQL е идеален както за малки, така и за големи приложения. Python in MySQL Python може да се използва в приложения за бази данни. Една от най-популярните бази данни е MySQL. MySQL база данни .За да можете да експериментирате с примерите за код в този урок, трябва да имате инсталиран MySQL на вашия компютър. Можете да изтеглите MySQL база данни от https://www.mysql.com/downloads/ .

# Инсталирайте MySQL драйвер:

Python се нуждае от MySQL драйвер за достъп до MySQL базата данни. В този урок ще използваме драйвера "MySQL Connector". Препоръчваме ви да използвате PIP, за да инсталирате "MySQL Connector". PIP най-вероятно вече е инсталиран във вашата Python среда. Навигирайте в командния ред до местоположението на PIP и въведете следното:

Изтеглете и инсталирайте "MySQL Connector":

# Python и MYSQL:

Python е популярен език за програмиране на високо ниво, известен със своя прост синтаксис, четливост и многофункционалност. Той се използва широко в различни приложения като уеб разработка, анализ на данни, машинно обучение, изкуствен интелект и автоматизация, наред с други.

MySQL е система за управление на релационни бази данни (RDBMS) с отворен код, който се използва широко за ефективно управление на големи количества данни. Той е известен със своята умственост, надеждност и производителност. MySQL се използва в различни приложения като електронна търговия, финанси, здравеопазване и социални медии, наред с други.

В този пример се свързваме с MySQL базата данни на локалната машина и изпълняваме простата заявка SELECT, за да извлечем всички данни от таблицата „клиенти“. Накрая отпечатваме резултатите.

# Python в MYSQL:

Python може да взаимодейства с MySQL бази данни, използвайки MySQL Connector/Python модул. Модулът MySQL Connector/Python предоставя API за работа с MySQL бази данни от код на Python. Можете да използвате този модул, за да се свържете с MySQL база данни, да изпълнявате SQL заявки и да манипулирате данни.

За да използвате модула MySQL Connector/Python, първо трябва да го инсталирате с помощта на мениджъра на пакети pip. Веднъж инсталиран, можете да импортирате модула във вашия код на Python и да използвате неговите функции за работа с базата данни MySQL.

Картина, която съдържа текст

Описанието е генерирано автоматичноЕто примерен кодов фрагмент, който демонстрира как да се свържете с MySQL база данни с помощта на Python и модула MySQL Connector/Python:

В този пример ние се свързваме с MySQL база данни на локалната машина и изпълняваме проста заявка SELECT, за да извлечем всички данни от таблицата „клиенти“. Накрая отпечатваме резултатите.

Python предоставя няколко други модула за работа с MySQL бази данни, като PyMySQL, SQLAlchemy и Django ORM. Тези модули предоставят по-разширени функции като обектно-релационно картографиране (ORM), управление на транзакции и миграции на бази данни. В зависимост от вашите нужди можете да изберете модула, който най-добре отговаря на вашите изисквания.

# База данни MYSQL:

MySQL е популярна система за управление на релационни бази данни (RDBMS) с отворен код, която използва Structured Query Language (SQL) за управление и манипулиране на данни, съхранявани в таблици. Първоначално е разработен от MySQL AB и сега е собственост на Oracle Corporation. MySQL се използва широко в уеб приложения и е известен със своята надеждност, производителност, мащабност за използване и лекота на използване.

MySQL поддържа широк набор от платформи, включително Windows, Linux и macOS, и може да се използва с много езици за програмиране, като PHP, Java, Python и Ruby. Той също така предлага различни инструменти и помощни програми за администрация на бази данни, като MySQL Workbench, MySQL Shell и MySQL Enterprise Monitor.

Някои от основните характеристики на MySQL включват:

Поддръжка за множество машини за съхранение, включително InnoDB, MyISAM и Memory.

Съответствие с ACID, което гарантира целостта и последователността на данните.  
Поддръжка за транзакции, което позволява множество операции с база данни да бъдат групирани в една транзакция.

Мащабност за използване, която позволява на MySQL да се справя с голям обеми данни и потребители.

Функции за сигурност, като удостоверение и криптиране.

Възможности за архивиране и възстановяване, които позволяват архивиране и възстановяване на данни в случай на загуба или повреда на данни.

Висока наличност, която позволява репликация и групиране, за да се осигури постоянна наличност на базата данни.

MySQL се използва широко в различни индустрии, включително електронна търговия, социални медии, финанси и здравеопазване. Неговата популярност може да се дължи на неговата лекота на използване, мащабност за използване и ефективните на разходи.

# FTP сървър:

FTP сървърът е компютърен [софтуер](https://en.wikipedia.org/wiki/Computer)[,](https://en.wikipedia.org/wiki/Software) състоящ се от една или повече [програми](https://en.wikipedia.org/wiki/Computer_program) , които могат да изпълняват команди, дадени от отдалечени [клиенти](https://en.wikipedia.org/wiki/Client_(computing)) , като получаване,изпращане, изтриване на [файлове](https://en.wikipedia.org/wiki/Computer_file) , създаване или премахване на [директории](https://en.wikipedia.org/wiki/Directory_(computing)) и т.н. Софтуерът може да работи като [софтуерен компонент](https://en.wikipedia.org/wiki/Software_component) на програма, като самостоятелна програма или дори като един или повече [процеси](https://en.wikipedia.org/wiki/Process_(computing)) (на заден план).

FTP сървърът играе ролята на [сървър](https://en.wikipedia.org/wiki/Server_(computing)) в [модела клиент-сървър,](https://en.wikipedia.org/wiki/Client%E2%80%93server_model) използвайки [FTP](https://en.wikipedia.org/wiki/File_Transfer_Protocol) и/или [FTPS](https://en.wikipedia.org/wiki/FTPS) и/или [SFTP](https://en.wikipedia.org/wiki/SSH_File_Transfer_Protocol)мрежови протоколи.

FTP сървърът може също да бъде предвиден като компютър , който изпълнява програма за FTP сървър за хостване на колекции от файлове. Големите FTP сайтове могат да се управляват от много компютри, за да могат да обслужват желания максимален брой клиенти, свързани към сървъри.

Клиентска програма се свързва с FTP сървър, след което, освен ако не е разрешен анонимен достъп, тя трябва да се удостовери чрез изпращане на [потребителско име](https://en.wikipedia.org/wiki/Username) и [парола](https://en.wikipedia.org/wiki/Password) ; след това може да извлича и/или изпраща файлове на сървъра заедно с други операции (в зависимост от привилегиите на потребителя).

## Определение на FTP сървър:

Сървърът за протокол за прехвърляне на файлове (известен като FTP сървър) е компютърен софтуер, който улеснява защитения обмен на файлове през TCP/IP мрежа. Той изпълнява протокола за прехвърляне на файлове (FTP), стандартен комуникационен протокол, който работи на ниво мрежа, за да установи защитена връзка между устройствата в архитектура клиент-сървър и ефективно да предава данни по интернет.

## За какво се използва FTP сървър?

FTP сървърите са софтуерни решения, използвани за прехвърляне на файлове през интернет. Те се използват предимно за две основни функции, „Поставяне“ и „Вземане“. Той позволява качване (Put) на файлове на сървъра от клиентското устройство и изтегляне (Get) на файлове от сървъра на клиентското устройство. FTP сървърът помага за изпълнението на следните функции.

## ****Обмен на файлове с голям размер:****

Организациите обикновено се затрудняват да споделят големи файлове по имейл. Бизнесите, работещи с огромни количества данни, често се сблъскват с прекъсвания по време на процеса на споделяне на файлове поради големи файлове. FTP сървърът позволява на организациите да споделят големи файлове без проблеми.

## ****Подобряване на сигурността:****

Най-важната цел на използването на FTP сървъри е да се осигури високо ниво на сигурност при изпращане на чувствителни данни през мрежата. FTP сървърите поддържат и други типове сигурни протоколи за прехвърляне на файлове, като SSH File Transfer Protocol (SFTP) и FTP Secure (FTPS), за да добавят още едно ниво на сигурност. Тези протоколи осигуряват ефективно криптиране от край до край за защита на файлове по време на пренос.

## ****Оптимизиране на работните потоци:****

FTP сървърите помагат на предприятията да рационализират процеса на споделяне на файлове, за да преодолеят предизвикателствата, свързани с продуктивността. С правилното софтуерно приложение на място, потребителите могат да споделят големи обеми от данни, вместо да споделят един файл наведнъж. Централното съхраняване на файлове минимизира времето, необходимо за намиране на файл, а планираните трансфери помагат да се избегнат забавяния или прекъсвания в работните потоци.

## ****Подобряване на контрола:****

FTP сървърите дават възможност на бизнеса да упражнява по-голям контрол върху своите данни, като предоставя интелигентни контроли за достъп. Тъй като всеки потребител изисква различни разрешения за достъп до различни файлове, администраторите могат лесно да определят кой може да редактира, качва, изтегля или споделя файлове въз основа на разрешенията.

## ****Надеждно възстановяване след бедствие:****

Един ефективен FTP сървър гарантира, че организационните данни и файлове не са компрометирани или загубени след бедствие. Непрекъснатото и автоматично архивиране помага за проактивно съхраняване на данни на други места за лесно възстановяване, когато е необходимо.

## Какви са видовете FTP?

[FTP Secure (FTPS):](https://www.serv-u.com/what-is-file-transfer-protocol-secure) Повишавайки сигурността с едно ниво от традиционния FTP, FTP Secure (FTPS) гарантира сигурно прехвърляне на файлове. Той осигурява допълнителен слой за криптиране, използвайки протоколи Secure Sockets Layer (SSL) или Transport Layer Security (TLS) по време на предаване на данни в мрежата.

[SSH протокол за прехвърляне на файлове (SFTP):](https://www.serv-u.com/what-is-ssh-file-transfer-protocol) Това е защитена подсистема за прехвърляне на файлове за протокола Secure Shell (SSH). SFTP е широко използван метод за сигурно прехвърляне на файлове през отдалечени системи. В SFTP както данните, така и командите се криптират и предават в специално форматирани двоични пакети чрез една защитена връзка, използваща SSH.

FTP активен срещу пасивен

FTP сесиите обикновено имат два канала, команден (контролен) канал и канал за данни. Докато командният канал се използва за предаване на команди, каналът за данни се използва за прехвърляне на данни. Администраторът може да постави FTP сървър в два режима, [активен режим и пасивен режим](https://www.serv-u.com/resource/tutorial/pasv-response-epsv-port-pbsz-rein-ftp-command) .

## ****Активен режим:****

Първоначално това беше режимът по подразбиране за FTP и различни сървъри все още го поддържат. В този режим FTP клиентът осъществява контролна връзка; въпреки това всички връзки за данни се инициират от сървъра към FTP клиента. Активният режим работи при липса на защитни стени или изисква защитни стени, които разбират FTP протокола, за да отварят автоматично портове между клиенти и сървъри. Известен е като активен режим, защото клиентът динамично отваря порт и слуша, докато сървърът активно се свързва с него. Препоръчва се само когато го изисква наследено внедряване.

## ****Пасивен режим:****

И двете връзки за данни и контрол се инициират от FTP клиента към FTP сървъра в този режим. Известен е още като режим „удобен за защитна стена“, тъй като работи в среда с инсталирани необходимите защитни стени. Известен е като пасивен режим, защото сървърът отваря порт и слуша пасивно, позволявайки на клиентите да се свързват с него. Пасивният режим се препоръчва за прехвърляне на файлове, тъй като връзките в този режим са по-сигурни и надеждни, тъй като връзките за данни се задействат от FTP клиента към FTP сървъра. Второ, не се допускат входящи връзки, започващи от интернет обратно към отделни клиенти. За разлика от активния режим, който изисква конфигуриране на множество защитни стени, пасивният режим изисква конфигуриране само на защитната стена на сървъра.

## FTP срещу облачно съхранение:

FTP и облак са двата начина за споделяне на файлове и данни и отчитат справедливия им дял от прилики и разлики. Докато FTP позволява прехвърляне на файлове между устройства в мрежа, облакът улеснява достъпа до съхранени данни и набор от услуги, включително изчислителни, мрежови и други, хоствани в интернет чрез уеб браузър или настолен клиент. Някои от основните разлики между FTP и облачно съхранение включват:

## ****Достъпност:****

Достъпът до файлове и папки в FTP настройка изисква използване на FTP клиент и задаване на необходимите разрешения за достъп до сървъра. Достъпът до данни, съхранени в облак, често изисква само уеб браузър или приложение, без да е необходимо потребителско име и парола. Въпреки това, въпреки че не е изискване, все още можете да съхранявате данни в облака зад защита с парола.

## ****Сигурност:****

Сигурността е проблем за организациите, използващи FTP. Има случаи на уязвимости на защитната стена, тъй като FTP връзката изисква отваряне на допълнителни портове за адекватна достъпност. Организациите, които искат да споделят файлове с други, може да се наложи да разширят достъпа до своята инфраструктура, увеличавайки риска за сигурността на физическия сървър. Освен това, липсата на възможност за проследяване създава повече пропуски, тъй като няма начин да се провери кой до каква информация е имал достъп. И накрая, организациите може да имат нужда от ИТ специалист, който периодично да архивира моментни снимки на сървъра на друго място. Като алтернатива, облачните услуги са напреднали с динамични контроли за сигурност, за да минимизират рисковете и уязвимостите за киберсигурността.

# Библиотеки за комуникация между сървър и клиент:

Python има много библиотеки, които могат да се използват за комуникация между сървър и клиент. Ето няколко често използвани библиотеки:

## Socket:

Това е мрежова библиотека от ниско ниво, която ви позволява да изпращате и получавате данни през мрежата. Той е относително лесен за използване, но трябва сами да се справите с всички подробности на мрежовия протокол.

## Twisted:

Това е по-усъвършенствана мрежова библиотека, която предоставя API на високо ниво за изграждане на мрежови приложения. Той е изграден върху рамка на Python, която е много подходяща за асинхронно програмиране.

## Flask:

Това е уеб рамка, която може да се използва за изграждане на уеб приложения, които комуникират със сървъра. Той предоставя API на високо ниво за обработка на HTTP заявки и отговори, и може да се използва за изграждане на RESTful API.

## Django:

Това е друга уеб рамка, която може да се използва за изграждане на уеб приложения. Той предоставя по-изчерпателен набор от функции от Flask, включително ORM за достъп до база данни, удостоверяване и администраторски интерфейси.

Socket е мрежова библиотека от ниско ниво в Python, която ви позволява да изпращате и получавате данни през мрежата. Той предоставя набор от функции и класове, които могат да се използват за създаване и управление на мрежови сокети, които са крайни точки, позволяват комуникация между две устройства по мрежата.

Със сокети можете да създавате клиент-сървър приложения, които могат да комуникират по различни мрежови протоколи като TCP, UDP и други. Библиотеката има прост и интуитивен API, който ви позволява лесно да установявате връзка между сървър и клиент, и да изпращате и получавате данни през връзката.

Ето някои ключови концепции, които трябва да имате предвид, когато работите със сокети:

## Типове сокети:

В Python има два типа сокети: TCP сокети и UDP сокети. TCP сокетите осигуряват надеждна, насочена към потокова комуникация, докато UDP сокетите са ненадеждни, насочени към дейтаграмно ориентирана комуникация.

## **IP адреси и портове**:

Когато установите връзка между клиент и сървър, трябва да знаете IP адреса и номера на порта на сървъра. IP адресът идентифицира устройството в мрежата, докато номерът на порта идентифицира конкретното приложение, работещо на това устройство.

Методи на сокет: Socket предоставя различни методи за създаване, обвързване, свързване, изпращане и получаване на данни през сокети. Тези методи включват socket(), bind(), listen(), accept(), connect(), send() и recv().

Обработка на грешки: Когато работите със сокети, е важно да се справите с грешките, които могат да възникнат по време на процеса на комуникация, като грешки при свързване или грешки при предаване на данни.

Като цялата библиотека на сокетите е мощен инструмент за изграждане на мрежови приложения в Python. Въпреки това определено изисква ниво на познания относно мрежовите протоколи и ниско ниво на мрежовите концепции, така че може да не е най-добрият избор за начинаещи или тези, които не са запознати с работата в мрежата.

Мрежовият протокол е набор от правила и конвенции, които управляват комуникацията между устройства в мрежа. Тези протоколи гарантират, че данните се предават надеждно и ефективно по мрежата и че устройствата могат да комуникират помежду си, независимо от техните хардуерни или софтуерни конфигурации.

Има много различни мрежови протоколи, всеки от които е предназначен за специфична цел или приложение. Ето някои често срещани примери:

## **Протокол за контрол на предаването (TCP**):

TCP е надежден, ориентиран към свързване протокол, който осигурява виртуална верига между две устройства. Той гарантира доставката на всички данни, изпратени по връзката, и гарантира, че данните се получават в правилния ред.

## Протокол за потребителски дейтаграми (UDP):

UDP е ненадежден протокол без връзка, който изпраща данни като поредица от дейтаграми. Той не гарантира доставка или ред на данни, но често се използва в ситуации, в които скоростта е по-важна от надеждността, като видео стрийминг или онлайн игри.

## Интернет протокол (IP):

IP е основният протокол, използван за маршрутизиране на данни между устройства в интернет. Той предоставя стандартна схема за адресиране, която позволява на устройствата да комуникират помежду си в различни мрежи.

## Протокол за прехвърляне на хипертекст (HTTP):

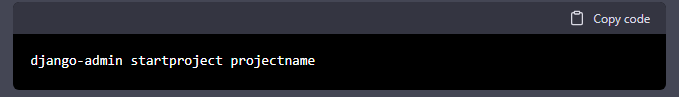
HTTP е протоколът, използван за прехвърляне на данни през World Wide Web. Той дефинира набор от методи за заявка (като GET и POST), които клиентите могат да използват, за да поискат ресурси от сървъри, и набор от кодове за отговор (като 200 OK и 404 Not Found), които сървърите използват, за да покажат състоянието на искания ресурс.

Използват се и много други мрежови протоколи, включително FTP, SMTP, SSH и много други. Всеки протокол има свой собствен набор от функции и характеристики, и изборът на правилния протокол за дадено приложение е важна част от проектирането на мрежова система.

# Сървър на django:

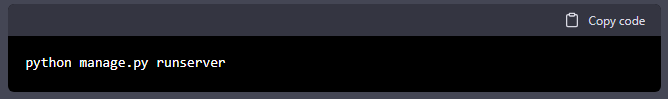
Django е високо ниво уеб рамка на Python, която позволява бързо и ефективно разработване на уеб приложения. Той следва модела на архитектурата Model-View-Controller (MVC), където моделът дефинира структурата на данните, изгледът управлява потребителския интерфейс, а контролерът управлява логиката.

За да стартирате Django сървър, първо трябва да инсталирате Django с помощта на pip (мениджър на пакети на Python). Веднъж инсталиран, можете да създадете нов Django проект с помощта на командата django-admin:



Това ще създаде нова директория с посоченото име на проекта и основна структура на проекта Django.

За да стартирате сървъра за разработка, отидете до директорията на проекта и изпълнете следната команда:



Това ще стартира сървъра за разработка на порт 8000 по подразбиране. Можете да получите достъп до приложението, като отворите http://localhost:8000 във вашия уеб браузър.

В Django дефинирате URL адреси, изгледи и шаблони за обработка на входящи заявки и генериране на отговори. URL адресите се съпоставят с изгледи, които са функции на Python, които обработват заявката и връщат отговор. Шаблоните определят HTML структурата на отговора.

Django също така предоставя мощна система за Object-Relational Mapping (ORM), която ви позволява да дефинирате вашите модели на база данни като класове на Python. ORM се грижи за картографирането на тези класове към таблиците на базата данни и предоставя удобен интерфейс за заявки и манипулиране на данни.

Като цяло Django е мощна и гъвкава уеб рамка, която улеснява изграждането на сложни уеб приложения с Python.

# Шини. Синхронен и асинхронен обмен:

Множеството крупни функционални елементи в структурата на всяка компютърна система се нуждае от обмен на информация за да функционира. За този именно обмен е необходима подходяща система от връзки (съединения). Системата от връзки трябва да осигурява обмен на информация между:

* Процесорните елементи и паметта ;
* Процесорните елементи и елементите на входно-изходната система ;
* Системата на паметта и елементите на входно-изходната система.

С развитието на компютърните системи се развиват и системите за връзка с отделните елементи в структурата и с външния свят. Повечето компютърни системи съдържат няколко различни шини, всяка от които е оптимизирана към определен вид комуникация. Част от шините е скрита във вътрешността на интегралните схеми или е достъпна само в рамките на печатната платка. Има шини, които са достъпни отвън с цел подключване на допълнителни устройства. Обикновено тези шини са стандартизирани, което позволява подключване на еднотипни устройства произведени от различни производители. За да характеризира дадена шина, е необходимо да се знае:

* Съвкупността на сигналните линии ;
* Физическите, механическите и електрическите характеристики на шината ;
* Съвкупността от сигнали за синхронизация, за управление, за състояние, за арбитриране ;
* Правилата за взаимодействие на присъединените към шината устройства (протокол на шината).

Операциите върху шината се наричат транзакции. Основните видове транзакции са транзакция за четене и транзакция за запис, или още транзакция за въвеждане или за извеждане (когато става дума за външни устройства). Шинната транзакция включва две противоположни по посока съставящи – изпращане на адрес и приемане на данни.

      Когато две устройства обменят информация посредством дадена шина, едно от тях инициира и управлява обмена. Такова устройство се нарича водещо (bus master). Водещо е всяко устройство, което може да заяви и в последствие да владее управлението на шината, т.е. да управлява трансфера на данни. Водещият не е задължен да използва данните сам. Той може да прави това в интерес на друго устройство. Устройствата, които не могат да инициират транзакции, се именуват водени (bus slave). Тъй като към шината могат да бъдат присъединени няколко водещи устройства, тя има магистрална организация, нещо което ще поясним малко по-късно. За получаване на правото да се владее, т.е. да се управлява шината, водещите устройства се подлагат на арбитриране.

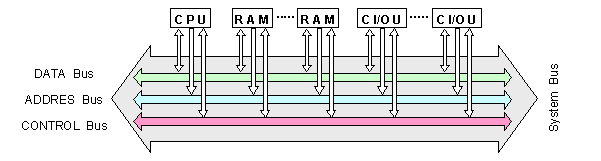
      Целевото предназначение на дадена шина определя нейните параметри. Така се класифицират три типа шини:

* Шини “процесор-памет" ;
* Шини за вход-изход ;
* Системни (управляващи) шини.

      В съвременните микропроцесори ефективността налага обменът между процесора и оперативната памет да се осъществява по независима шина. Тази шина се нарича предна шина FSB (Front-Side Bus). В групата на шините "процесор-памет” се включва и шината за връзка с кеш паметта. Тук става дума за задна или вторична шина BSB (Back-Side Bus). Независимият обмен на процесора с оперативната памет и с кеш паметта обединява FSB и BSB в архитектура, известна като DIB (Dual Independent Bus).

      Шините за вход-изход свързват процесора (паметта) с входно-изходните устройства. Отчитайки разнообразието на тези устройства, шините за вход-изход се унифицират и стандартизират. Поради по-ниските скорости за обмен с болшинството външни устройства от шините за В/И не се изисква висока пропускателна способност. При конструиране на тези шини се преследва ниска цена и в същото време възможно по-голяма дължина на линиите.

      С цел на се намали общата цена на компютърната система много от тях се проектират с обща шина за памет и за Б/И устройства. Такива шини често се наричат системни. Системната шина служи (често наричана backplane bus) за физическо и логическо обединяване на всички устройства.



Понятие и определения за шина

      Вече многократно ни се наложи да употребяваме понятието шина. Тук това понятие употребяваме в смисъла на разглежданата тематика. В този контекст това не е просто логическа връзка между два структурни елемента, а връзка за реализация на трансфер с магистрална организация. Понятието шина е събирателно, твърде обобщено. В нейния състав влизат отделни шини (групи съединителни линии), разпространяващи специфична информация, в смисъла на която формират:

1.       Информационен канал ;

2.       Магистрала за управление на информационния канал.

По информационния канал се предават:

1.       Кодове на данни ;

2.       Кодове на адреси ;

3.       Кодове на команди ;

4.       Кодове на състоянията.

      Информационният канал съдържа адресна шина и даннова шина, а магистралата за управление на информационния канал съдържа:

* Шина за предаване на управлението (линии свързани с арбитража) ;
* Шина за управление на обмена (синхронизация) ;
* Шина за прекъсванията ;
* Шина на специални сигнали (контрол на захранването, общ сигнал Reset, контролни кодове за откриване и поправка на грешки при предаване и други задължителни и незадължителни линии).

      Данновата шина обикновено се разбира като паралелна (има определена ширина, която се измерва в битове). Тя е основна шина и е двупосочна, тъй като реализира обмен във двете посоки. Подключването на устройствата към данновата шина се осъществява чрез буфери с три състояния. Последното се налага поради нейната двупосочна и магистрална организация. Ширината на данновата шина е най-съществената характеристика и определя (именова) дадена шина като например, 32-битова. Освен това определя редица други технически характеристики като например, максималната скорост за трансфер на данни.

      Втора по значение е адресната шина. Нейната ширина е свързана с дължината на адреса, който използва даденият процесор. От гледна точка на процесора тази шина е еднопосочна. Възможно е тази шина да се използва и за трансфер на друга информация, например, данни. В такива случаи тя се определя като мултиплексна, т.е. като шина, която превключва своето предназначение. Основната причина да се прави това, е стремеж към минимизиране на физическите връзки, а оттам поевтиняване на системата като цяло.

      Управляващата шина е помощна. Нейният състав при различните интерфейсни шини е твърде различен. Общо е обаче разбирането, че в нейния състав са включени онези линии, които разпространяват управляващите сигнали, като код на операцията, сигнали за разрешение, за валидност, признаци за състояние и за грешки и други специфични за концепцията на шината стойности. Отделните линии в управляващата шина могат да бъдат едно или двупосочни. За тези линии от особено значение е новото на активната логическа стойност. Говорим, че управлението е по нулата или по единицата. Основното предназначение на управляващата шина е да осъществи и контролира синхронизацията между обменящите се устройства. Особено значение за надеждността на процеса на обмен имат фронтовете на управляващите сигнали. Моментите, в които се появяват тези фронтове маркират и реализират във времето съответния шинен протокол, или казано с други думи, реализират в крайна сметка фиксирането на данните по предназначение. Сигналите, които непосредствено извършват това, се наричат строб-сигнали. Строб сигналите могат да се определят още като строб за четене и строб за запис.

      В зависимост от протокола, т.е. от алгоритъма, по който се реализира обменът по шината, последният се определя цялостно като:

* Синхронен. Обменът, а от там и шината, се определят като синхронни, когато инициаторът на обмена (процесор, контролер) приключва обмена в рамките на установен от него времеви интервал, т.е.без да се интересува от интересите на изпълнителното (подчиненото или воденото) устройство ;
* Асинхронен. Този обмен и такава шина се определят като асинхронни, когато инициаторът на обмена е в диалогова връзка с подчиненото устройство. Инициаторът приключва (завършва) обмена след като получи от изпълнителя специален сигнал потвърждение.

## Асинхронен обмен:

      Асинхронният метод за организация се постига с помощта на две групи сигнали - управляващи и оповестяващи. Тези две групи сигнали свързват инициатора и изпълнителя и са част от шината за обмен. Процесът на обмен е своеобразен диалог, който се провежда между свързалите се по шината устройства и също се нарича протокол.

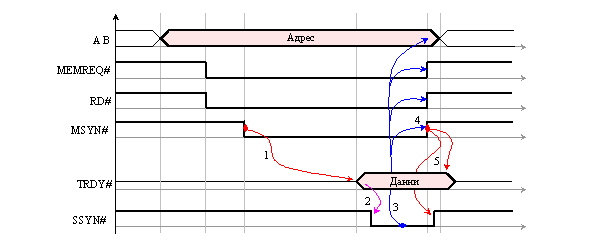
      Основното достойнство на асинхронния обмен се изразява в по-високата надеждност при предаване на данните. Протоколът за обмен е изключително гъвкав и адаптируем, така че е в състояние да организира и осъществи обмен между устройства със силно различаващи се скоростни показатели. Недостатъкът се изразява главно в големия брой сигнали. Колкото по-прецизен е алгоритъмът на диалога и колкото повече различни ситуации са вградени в него, толкова повече са сигналите на шината.

      Ще поясним асинхронния обмен на примерна времедиаграма върху шина, в чийто състав влизат:

* АВ  –  Адресна шина. По нея се предава адрес и команда към подчиненото устройство ;
* DB  –  Даннова шина ;
* MEMREQ#  –  (Memory Request) заявка за обмен с паметта ;

·         RD#  –  (Read) код на операцията четене, която ще изпълни подчиненото устройство ;

* MSYN#  –  (Master Synchronization) сигнал за начало на обмена ;
* SSYN#  –  (Slave Synchronization) сигнал отговор от подчиненото устройство.



Инициаторът на обмена издава адрес и го поддържа по време на трансфера, след което изпраща с необходимите закъснения заявката до нужното изпълнително устройство заедно с кода на операцията. Тези, както и други възможни действия от страна на инициатора, подготвят обмена и след като подготовката завърши, издава сигнала MSYN# .

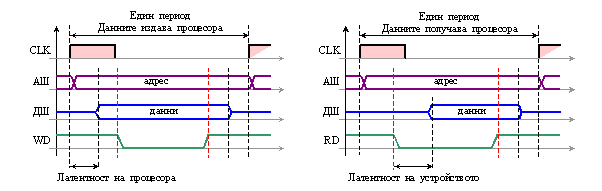
      В отговор на този сигнал подчиненото устройство започва да изпълнява операцията. Разбира се, това то прави с присъщата си скорост. Когато данните са извлечени и заредени в изходния буфер, заедно с тях устройството издава синхронизиращия сигнал SSYN#. Инициаторът, който до момента е бил в очакване, получавайки този сигнал, разбира, че данните са върху шината и ги фиксира в своя буфер. Данните в представената времедиаграма следва да се разбират обобщено – това може да е една порция, но може да бъде и блок. Трансферът на блок обаче изисква допълнителни синхронизиращи сигнали, които не са показани. Трансферът по принцип се прекратява от инициатора чрез снемане на сигнала MSYN# .

      Когато водещото устройство фиксира успешно в своя буфер получените данни, то снема от шината сигнала MSYN#. Заедно с този сигнал инициаторът прекратява (в съответния ред) подаването и на останалата информация – адрес, команда, избор на устройство. В същото време, за подчиненото устройство изчезването на този сигнал означава, че данните са достигнали успешно предназначението си и трансферът ще бъде закрит. Така в отговор подчиненото устройство снема от шината активната стойност на сигнала SSYN# и изключва своите изходни буфери.

      Асинхронният обмен се основава на причинно-следствените връзки и се нарича още обмен с квитиране. Повечето съвременни шини осъществяват обмен, в който прилагат оптимална смес от принципите на синхронния и на асинхронния протокол.

## Синхронен обмен:

      Ще се спрем най-напред на синхронния обмен. Синхронизацията се постига чрез специален периодичен правоъгълен сигнал с определена честота, определяща темпа на работа на всички устройства, които го използват. В общият случай в трансфера вземат участие следните елементи – адрес, данни, строб на кода на операцията и сигнали на допълнителните условия. В зависимост от това кой е източникът на данните и какво е неговото закъснение при генериране на данните са възможни следните времедиаграми:

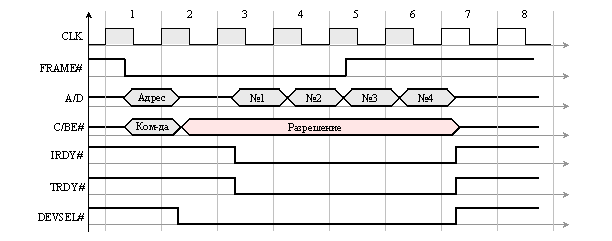


 Данните, излизащи от процесора се съпровождат от строб-сигнала за запис WD. За надеждност на записа процесорът продължава да поддържа данните в рамките на текущия цикъл и след изчезване на записващият строб. При обратния обмен данните подава външно устройство (контролер), което обикновено има по-голяма латентност. В отговор на строб-сигнала за четене RD, който процесорът издава, устройството поставя данните върху шината, а в отговор на изчезването на строб-сигнала – ги снема от шината. И в двата случая данните изчезват от шината след стробиращите сигнали.

      Освен еднотактен цикълът на синхронен обмен може да бъде многотактен. Многотактните цикли могат да бъдат с фиксирана и неограничена продължителност. Ще дадем още един пример за синхронен обмен въз основа на най-разпространената към момента системна шина по стандарта PCI. Цикълът на обмен по тази шина не е фиксиран по продължителност и се нарича транзакция. Един период на синхросигнала в транзакцията се нарича фаза. Ще ни бъдат необходими още следните означения:

* CLK  –  Синхронизираща последователност на шината (от 33 до 66[MHz]) ;
* FRAME#  -  Сигнал активна нула, издаван от устройството инициатор на трансфера, определящ продължителността на транзакцията ;
* A/D  –  32 или 64 битова шина за предаване на адрес и данни. Шината е мултиплексна ;
* C/BE#  -  Представлява шина от 4 линии за команда (С) и за масив байтове (ByteEnable#) ;
* IRDY#, TRDY#  -  Сигнали, показващи готовността на устройството инициатор и на целевото устройство ;
* DEVSEL#  -  Сигнал отговор на целевото устройство за това, че разпознало своя адрес и е готово за трансферната операция ;
* IDSEL#  -  Сигнал за избор на устройство след конфигуриране от тип CS (Chip Select) .

      Основно положение в PCI-шината е, че адресът се предава в началото на транзакцията и се съхранява в буферен регистър в целевото устройство. Така в последствие линиите на шината A/D се използват за трансфера на данните. На следващата фигура е изобразен трансфер на 4 последователни порции данни. Всички превключвания по шината са синхронизирани по предния фронт на сигнала CLK. Допустимите конкретни стойности на закъсненията, както и останалите сигнали, които са дефинирани за PCI-шината, читателят може да узнае от пълното описание на стандарта и неговите модификации. За нас тук сега те са принципно без значение.



 Представената транзакция на 4 последователни порции данни (примерно 4х4 байта) отнема 6 периода (6 такта или още 6 фази). В първата фаза инициаторът на трансфера генерира сигнала FRAME# и заедно с него поставя на шината 32-битов адрес и 4-битова команда. Командата е за операция четене и се отнася за оперативната памет. Оперативната памет при тези сигнали следва да съхрани изпратения адрес за да го използва за организиране на последователния достъп, а той ще продължи, докато не изчезна разрешението, което се явява по шината C/BE# веднага след командата във втора фаза. Инициаторът използва втория такт за да превключи шината A/D като входна, тъй като по нея ще приеме изпратените данни. Докато инициаторът изключва за да пренасочи входно-изходните буфери на шината A/D, целевото устройство включва своите изходни буфери и установява сигнала DEVSEL#, с което оповестява, че е получило командата за трансфер.

      По време на трансфера инициаторът поддържа активни определен брой линии от шината C/BE#. Логиката на това поддържане се съдържа във възможностите на целевото устройство. Ако то има 32 битова изходна шина, тогава инициаторът поддържа активни и четирите линии BE#, в противен случай само възможния брой.

      В началото на третия такт инициаторът активира сигнала IRDY#, с което съобщава за собствената си готовност да приема данни. В същото време целевото устройство активира своето потвърждение чрез сигнала TRDY#, като заедно с това поставя порция данни №1 върху шината. В края на такта инициаторът фиксира получените данни в своя буфер. Така в началото на всяка следваща фаза целевото устройство издава поредната порция данни, а инициаторът я приема в края на фазата. За оповестяване края на трансфера инициаторът използва сигнала FRAME#, чиято активна нула снема в предпоследния такт, който в дадения пример е 5-ти. В края на 6-ти такт инициаторът фиксира последната порция, след което по време на 7-ми такт както той, така и целевото устройство. възстановяват изходното си състояние и освобождават шината.

      Двустранните сигнали за готовност IRDY# и TRDY# могат да бъдат прекратявани по време на трансфера, ако някой от участниците загуби готовността си. Това може да бъде причина за удължаване на съответната фаза, без обаче да се счита, че трансферът е прекратен. Успешна е всяка фаза, по време на която и двата сигнала са активни.

      Главното достойнство на синхронния обмен се състои в простотата на протокола и минималния брой управляващи сигнали. Недостатъците се изразяват главно в липсата на диалог, т.е. в това, че отсъстват гаранции за това, че изпълнителят ще изпълни зададената операция. Освен това синхронизацията налага високи изисквания към бързодействието на изпълнителя.

# Въведение в XAMPP:

XAMPP е крос-платформен уеб сървър, който е безплатен и с отворен код. XAMPP е кратка форма за крос-платформа, Apache, MySQL, PHP и Perl. XAMPP е популярен междуплатформен уеб сървър, който позволява на програмистите да пишат и тестват своя код на локален уеб сървър. Той е създаден от Apache Friends и обществеността може да ревизира или модифицира неговия собствен изходен код. Той включва MariaDB, Apache HTTP сървър и интерпретатори за PHP и Perl, наред с други компютърни езици. Поради простотата на разгръщане на XAMPP, разработчикът може бързо и лесно да инсталира WAMP или LAMP стек на операционна система, с допълнителното предимство, че могат да се зареждат и общи приложения за добавки като WordPress и Joomla.

## Нужда от XAMPP:

* XAMPP е просто локален хост или сървър.
* Този локален сървър работи на вашия персонален компютър, независимо дали е настолен или лаптоп.
* Използва се за тестване на клиенти или уебсайтове, преди да ги публикува на отдалечен уеб сървър.
* На локален компютър сървърният софтуер XAMPP осигурява подходяща среда за тестване на MYSQL, PHP, Apache и Perl проекти. Тъй като повечето внедрявания на уеб сървъри в реалния свят споделят същите компоненти като XAMPP, преминаването от локален тестов сървър към сървър на живо е лесно.

## Предимства на XAMPP:

В сравнение с други уеб сървъри като WAMP, той е лесен за настройка.

Той е Multi Cross-Platform, което предполага, че работи както на Windows, така и на Linux.

С една команда можете да стартирате и спирате целия стек на уеб сървъра и базата данни.

Налични са както пълна, така и стандартна версия на XAMPP.

Той има контролен панел, който можете да видите, че съдържа бутони за стартиране и спиране за специфични механизми, като например Apache, който работи през своя контролен панел.

Той също така включва OpenSSL, phpMyAdmin, MediaWiki, Joomla, WordPress и много допълнителни модули.

## Недостатък на XAMPP:

В сравнение със сървъра WAMP конфигурацията и настройката са по-трудни.

## Компоненти на XAMPP:

Компонентите, които са включени в XAMPP, са дадени по-долу:

* Крос-платформа: Различни операционни системи се инсталират в отделни конфигурации на различни локални системи. Компонентът за различни платформи е включен, за да подобри функционалността и обхвата на този пакет за разпространение на Apache. Работи с различни платформи, включително Windows, Linus и MAC OS пакети.
* Apache: Apache е междуплатформен HTTP уеб сървър. Използва се за транспортиране на уеб материали по целия свят. Ако някой поиска файлове, снимки или документи, използвайки своя браузър, HTTP сървърите ще обслужват такива активи на клиентите.
* MariaDB база данни: XAMPP използва за включване на MySQL DBMS; обаче MariaDB вече зае мястото си. MySQL е една от най-широко използваните системи за управление на релационни бази данни. Той предоставя услуги за съхранение, манипулиране, извличане, управление и изтриване на данни чрез интернет.
* PHP: Пълната форма на PHP е Hypertext Preprocessor. PHP е backend език за програмиране, който най-често се използва в уеб разработката. Потребителите могат да използват PHP за изграждане на динамични уебсайтове и приложения. Той поддържа различни системи за управление на бази данни и може да бъде инсталиран на всяка платформа. Написан е на езика за програмиране C.
* Perl: Perl често се нарича "общ" език за програмиране. Този език Perl е динамичен и интерпретируем. Този език се използва за уеб разработка, разработка на GUI, системна администрация и други неща. HTML, XML и други езици за маркиране се поддържат от Perl.
* phpMyAdmin: Това е инструмент за администриране на база данни за MariaDB.
* OpenSSL: OpenSSL е реализация с отворен код на SSL и TLP.
* Контролен панел на XAMPP: Контролният панел на XAMPP е панел, който подпомага работата и регулирането на други компоненти на XAMPP.
* Webalizer: Това е софтуерно решение за уеб анализ, което следи потребителски регистрационни файлове и отчети за използването.
* Mercury: Това е пощенски сървър, който помага при управлението на имейли в интернет.
* Tomcat: Това е JAVA-базиран сървър, който осигурява JAVA функционалност.
* Filezilla: Това е сървър за протокол за прехвърляне на файлове (FTP сървър), който улеснява и поддържа процесите на прехвърляне на файлове.

## Приложения на XAMPP:

* Създателите на XAMPP възнамеряват да се използва като инструмент за разработка, позволяващ на уеб дизайнерите и програмистите да тестват работата си на персоналните си компютри без нужда от интернет връзки. Много ключови елементи за сигурност са деактивирани по подразбиране, за да направи това възможно най-лесно. XAMPP се използва за обслужване на уеб страниците в Интернет.
* Може също да се използва за създаване и манипулиране на бази данни в MariaDB и SQLite, наред с други бази данни.
* След като XAMPP бъде инсталиран, FTP клиент може да се свърже с локален хост и да го третира като отдалечен хост. Когато инсталирате система за управление на съдържание като Joomla или WordPress, използвайте инструмент като FileZilla. Можете също да използвате HTML редактор, за да се свържете с локален хост чрез FTP.

# Обяснение на кода:

from pyftpdlib.authorizers import DummyAuthorizer

from pyftpdlib.handlers import FTPHandler

from pyftpdlib.servers import FTPServer

# Define the FTP server settings

ftp\_port = 2121

ftp\_username = 'username'

ftp\_password = 'password'

ftp\_directory = '/12b/ftp'

# Define the user authorizer

authorizer = DummyAuthorizer()

authorizer.add\_user(ftp\_username, ftp\_password, ftp\_directory, perm='elradfmwMT')

# Define the FTP handler

handler = FTPHandler

handler.authorizer = authorizer

# Start the FTP server

server = FTPServer(('192.168.99.104', ftp\_port), handler)

server.serve\_forever()

* Кодът започва с дефиниране на настройките на FTP сървъра.
* Ftp\_port е зададен на 2121, а ftp\_username, ftp\_password и ftp\_directory са дефинирани като 'username', 'password' и '/12b/ftp'.
* След това се създава потребителски авторизатор с DummyAuthorizer().
* Този авторизатор ще позволи на всеки потребител, който е бил удостоверен с потребителско име='потребителско име' и парола='парола', за достъп до директория '/12b/ftp'.
* Следващата стъпка в кода е създаването на екземпляр на FTPHandler.
* Този манипулатор ще използва DummyAuthorizer() като свой авторизатор.
* Накрая се създава FTPServer обект с помощта на тези два обекта (FTPServer класът приема два аргумента: име на хост и манипулатор).
* След това методът serve\_forever() на този обект започва да работи вечно на порт 2121.
* Кодът е прост пример за това как да създадете FTP сървър с помощта на pyftpdlib.
* Първият ред декларира променливите, които ще бъдат използвани в кода, включително: - ftp\_port: Номерът на порта за FTP сървъра.
* - ftp\_username: Потребителското име за FTP сървъра.
* - ftp\_password: Паролата за FTP сървъра.
* - ftp\_directory: Директорията, където се съхраняват файлове на FTP сървъра.
* - perm: Низ, който определя какви разрешения трябва да бъдат предоставени на потребителите, които се свързват към този конкретен екземпляр на FTPServer обект.

# Използвана литература:

1. https://www.w3schools.com/python/python\_mysql\_getstarted.asp
2. https://www.datacamp.com/tutorial/mysql-python
3. https://bg.theastrologypage.com/ftp-server
4. https://bg.alegsaonline.com/art/20972
5. https://www.djangoproject.com/
6. http://www.tyanev.com/home.php?s=372&lang=bg&mid=18&mod=1&b=12
7. https://www.educba.com/what-is-xampp/
8. https://docs.python.org/3/library/socket.html
9. https://www.interviewbit.com/blog/flask-vs-django/
10. https://twisted.org/