Консольные приложения — используют для взаимодействия с пользователем консоль.

Консоль – интерфейс, который использует стандартный ввод и вывод данных.(это текстовый интерфейс, в котором команды отдаются путем ввода текстовых строк с клавиатуры.)

*Поток — это байтовая последовательность, передаваемая в про­цессе ввода-вывода.*

*Стандартные потоки — механизм, позволяющий осуществлять взаимодействие с пользователем как чтение и запись в файл.*

*Стандартные потоки назначаются и открываются системой автоматически.*

stdin — поток стандартного ввода (обычно связан с клавиатурой);

stdout — поток стандартного вывода (обычно связан с дисплеем);

stderr — вывод сообщений об ошибках (связан с диспле­ем).

- Сложны для пользователя (отсутствует интерфейс)

+ Возможность вывод одного приложения перенаправить в ввод другого приложения

+ Нересурсоёмкие

Конвеер = цепочка консольных приложений, так называемые скрипты, где перенаправлен вывод из одного приложения в ввод другого.

Результат выполнения можно выводить не только на экран.

Перенаправление осуществляется “>” перед символом можно указать номер дескриптора.

Десктопные приложения - прикладные приложения работающие в среде операционной системы. Приложение инсталлируется на рабочую станцию пользователя и запускается локально или удаленно.

+ Интерфейс

- Зависимость от ОС

- Ресурсоёмкие

- Нельзя выстраивать в цепочки

Инсталяция включает в себя:

* копирование файлов с носителя в память жесткого диска компьютера;
* проверку ПК на совместимость;
* размещение всей файлов в правильных папках и в правильной очередности;
* создание самостоятельно выполняющихся команд в реестре и автозагрузке;
* создание конфигурационных файлов, позволяющих менять настройки программы в дальнейшем по желанию пользователя.

Портативные программы - это приложения, которые будут полностью функционировать, без их установки на вашем компьютере. Другими словами, вам достаточно скопировать такое приложение в любой каталог, а затем запустить его. В отличии от других программ, такие приложения не хранят свои настройки или параметры в реестре Windows и не привязываются к определенной системе каким-либо образом.

**Клинт-серверное приложение: разделение клиент, сервер потому что появиля интернет для обеспечения совместной работы пользователей с данными.**

**Клиентская часть будет взаимодействовать с пользователем. Серверная часть обеспечивает хранение данных**

**Взаимодействие** между клиентским и серверным процессами представляет собой совместный транзакционный обмен, в котором активность исходит от **клиента**, а **сервер** реагирует на эту активность. Приложение **клиент**-**сервер** Любое приложение, в котором инициатор действия находится в одной системе, а исполнитель действия — в другой.

Плюсы:

Множество клиентов, которые могут подключаться к серверу.

Отсутсвие дублирования кода программы-сервера программами-клиентами

Все данные хранятся централизованно на сервере, который, как правило, защищён гораздо лучше большинства клиентов.

Безопасность – если случается ошибка на клиенте, данные на сервере все равно сохраняются.

Гарантия целостности данных. Сервер реализует управление транзакциями, и предотвращает попытки одновременного изменения одних и тех же данных.

На сервере проще организовать контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа

Минусы:

Неработоспособность сервера может сделать неработоспособной всю вычислительную сеть  
Поддержка работы данной системы требует отдельного специалиста — системного администратора  
Высокая стоимость оборудования

3х ур - Предполагает наличие в ней 3х компонентов - клиента, сервера приложений (к которому подключено клиентское приложение) и сервера баз данных (с которым работает сервер приложений)

2х ур - Предполагает наличие в ней 2х компонентов - клиента, сервера приложений (к которому подключено клиентское приложение)

В двухуровневой архитектуре клиент-сервер приходится распределять три основые части приложения (хранение данных, обработка данных, интерфейс пользователя) по двум физическим модулям. Обычно хранения данных располагается на сервере, интерфес пользователя на стороне клиента, а обработку приходится делить между клиентской и серверной частями, что может вызвать ряд сложностей. Для их решения используют многоуровневую(3 и более уровней) архитектуру клиент-сервер  
**многоуровневая архитектура** - разновидность архитектуры, в которой функция обработки данных вынесена на один или несколько отдельных серверов, выполняющие функции сервера для интерфейсов с пользователем и функции клиента для серверов баз данных

Клиент - Интерфейсный (обычно графический) компонент комплекса, предоставляемый конечному пользователю. На этот уровень обычно выносится только простейшая бизнес-логика: интерфейс авторизации, алгоритмы шифрования, проверка вводимых значений на допустимость и соответствие формату, несложные операции с данными (сортировка, группировка, подсчёт значений), уже загруженными на терминал.

Сервер приложений - Располагается на втором уровне, на нём сосредоточена большая часть бизнес-логики. Вне его остаются только фрагменты, экспортируемые на клиента, а также элементы логики, погруженные в базу данных.

Сервер базы данных - Обеспечивает хранение данных и выносится на отдельный уровень, реализуется, как правило, средствами систем управления базами данных, подключение к этому компоненту обеспечивается только с уровня сервера приложений.

**Сравнение 2х и 3х уровневых архитектур**

2х уровневая архитектура проще в реализации.

**Производительность**

Трехуровневая архитектура обеспечивает большую производительность (производительность одного конкретного запроса). Задачи можно распределить между разными серверами. Уменьшается нагрузка на сеть, так как передаются обработанные данные.

**Безопасность**

Трехуровневая архитектура обеспечивает большую безопасность. Больше звеньев, в каждом из которых можно определить проверки безопасности. У клиентов нет прямого доступа к данным.

**Масштабируемость**

Большая гибкость и масштабируемость. Каждый слой можно изменять отдельно, проще вносить изменения. Количество серверов на каждом слое можно увеличивать более гибким образом.

Тонкий клиент - Программа-клиент которая переносит большую часть задач по обработке информации на сервер.

- Узкий функционал

- Невозможность работы при обрывах связи с сервером

- Чаще всего более низкое быстродействие

+ Бизнес-логика находится на сервере (Большая безопасность), обновлять придётся только в одном месте.

Толстый клиент - Программа-клиент обеспечивающая расширенную функциональность независимо от центрального сервера. Часто сервер в этом случае является лишь хранилищем данных, а вся работа по обработке и представлению этих данных переносится на машину клиента.

- Перезагружается сеть, из-за передачи необработанных (а значит избыточных) данных  
- Усложняется поддержка и изменение системы, любое изменение требует замены всех интерфейсных программ

-Проблемы безопасности (возможно логин и пароль для подключения к бд лежит на клиенте.)  
- Наличие бизнес-логики -> (При обновлении бизнеслогики нужно обновить все клиенты).

+более высокое быстродействие

+ частично может работать без интернета.

**Типы серверов**

Web-сервера – Предоставляют доступ к веб страницам и файлам по протоколу http.

Серверы приложений – предоставляют площадку для запуска приложений, описывающих бизнес логику, написанных на каком-либо из языков программирования.

Серверы управления базами данных – для хранения данных и обработки запросов на выборку.

Файл-серверы – хранит информацию в виде файлов и предоставляют пользователям доступ к ним.

Почтовые сервера – предоставляют услуги по отправке и получению электронных почтовых сообщений.

Серверы сообщений – предоставляют услуги по отправке и получению мгновенных сообщений.

**Промежуточное ПО (Middleware)** – программное обеспечение, предназначенное для объединения компонентов клиент-серверного приложения, упрощения взаимодействия между ними.

* ПО обеспечения межпрограммного взаимодействия.

Вызов удаленных процедур (RPC).

* Сервисы обработки сообщений. (*MOM — message-oriented middleware*)

Обмен сообщениями реализуется через API системы MOM. Запросы сервисов ставятся в очередь сообщений и обрабатываются в соответствии с приоритетами и доступностью ресурсов. Надежная доставка сообщений (без потерь), гарантированная доставка сообщений, застрахованная доставка сообщений (каждое сообщение отправляется только один раз).

* Мониторы обработки транзакций (*Transaction Processing monitors*, TP-monitors)– промежуточное ПО, обеспечивающее контроль передачи данных от клиента при работе с распределенными бд (с множеством бд от различных поставщиков). Обеспечивает целостность данных, следя за тем, чтобы не было потерянных или незавершенных транзакций.

СЛОИ

* Данных
* Бизнес-логики – реализация предметной области. Задачи, для которых предназначено приложение.
* Предстваления