Служебное программное обеспечение.
Базы данных. СУБД.
План.

1. Назначение и классификация служебных программных средств.
2. Компьютерный вирус. Признаки вирусного заражения ЭВМ.
3.Классификация вирусов.
4. Виды антивирусных программ. Меры по защите ЭВМ от заражения вирусами.
5. Понятие архивации. Методы сжатия. Форматы сжатия.
6. Основные идеи алгоритмов RLE, Лемпеля-Зива, Хаффмана.
7. Базы данных. Классификация БД. Этапы создания БД.
8. СУБД. Основные компоненты и принципы создания СУБД.

В основу работы компьютеров положен программный принцип управления, состоящий в том, что компьютер выполняет действия по заранее заданной программе. Этот принцип обеспечивает универсальность использования компьютера: в определенный момент времени решается задача соответственно выбранной программе. После ее завершения в память загружается другая программа и т.д.

Программное и аппаратное обеспечение в компьютере работают в неразрывной связи и взаимодействии. Состав программного обеспечения вычислительной системы называется **программной конфигурацией.** Между программами существует взаимосвязь, то есть работа множества программ базируется на программах низшего уровня.

Междупрограммный интерфейс - это распределение программного обеспечения на несколько связанных между собою уровней. Уровни программного обеспечения представляют собой пирамиду, где каждый высший уровень базируется на программном обеспечении предшествующих уровней.

Структура программного обеспечения:

Прикладной уровень
Служебный уровень
Системный уровень
Базовый уровень

Служебный уровень

Программы этого уровня взаимодействуют как с программами базового уровня, так и с программами системного уровня.

Назначение служебных программ (утилит): *автоматизация работ по проверке и настройки компьютерной системы, а также для улучшения функций системных программ.*

Некоторые служебные программы (программы обслуживания) сразу входят в состав операционной системы, дополняя ее ядро, но большинство являются внешними программами и расширяют функции операционной системы. То есть, в разработке служебных программ отслеживаются два направления: интеграция с операционной системой и автономное функционирование.

Классификация служебных программных средств.

1. ***Диспетчеры файлов*** *(файловые менеджеры:*

Выполняют большинство операций по обслуживанию файловой структуры: копирование, перемещение, переименование файлов, создание каталогов (папок), уничтожение объектов, поиск файлов и навигация в файловой структуре.

2. ***Средства сжатия данных (архиваторы).***

Предназначены для создания архивов.

3. ***Средства диагностики.***

Предназначены для автоматизации процессов диагностики программного и аппаратного обеспечения. Их используют для исправления ошибок и для оптимизации работы компьютерной системы.

4. ***Программы инсталляции (установки).***

Предназначены для контроля за добавлением в текущую программную конфигурацию нового программного обеспечения.

5. ***Средства коммуникации****. Разрешают устанавливать соединение с удаленными компьютерами, передают сообщения электронной почты, пересылают факсимильные сообщения и т.п.*

6. ***Средства просмотра и воспроизведения.***

Преимущественно, для работы с файлами, их необходимо загрузить в "родную" прикладную программу и внести необходимые исправления. Но, если редактирование не нужно, существуют универсальные средства для просмотра (в случае текста) или воспроизведения (в случае звука или видео) данных.

7. ***Средства компьютерной безопасности.***

К ним относятся средства пассивной и активной защиты данных от повреждения, несанкционированного доступа, просмотра и изменения данных. Средства пассивной защиты - это служебные программы,

предназначенные для резервного копирования. Средства активной защиты применяют антивирусное программное обеспечение.

КОМПЬЮТЕРНЫЕ ВИРУСЫ И АНТИВИРУСНЫЕ ПРОГРАММЫ

Компьютерный вирус (КВ) *– это программа, способная создавать свои копии (не обязательно полностью совпадающие с оригиналом), внедрять их в различные объекты или ресурсы компьютерных систем, сетей и производить определенные действия без ведома пользователя.*

Свое название КВ получил за некоторое сходство с биологическим вирусом. Например, в зараженной программе самовоспроизводится другая программа-вирус, а инфицированная программа может длительное время работать без ошибок, как в стадии инкубации.

Программа, внутри которой находится вирус, называется зараженной (инфицированной) программой.

Когда инфицированная программа начинает работу, то сначала управление получает вирус. Он заражает другие программы, а также выполняет запланированные деструктивные действия. Для маскировки своих действий вирус активизируется не всегда, а лишь при выполнении определенных условий (истечение некоторого времени, выполнение определенного числа операций, наступление некоторой даты или дня недели и т.д.). После того, как вирус выполнит нужные ему действия, он передает управление той программе, в которой он находится. Внешне зараженная программа может работать так же, как и обычная программа. Подобно настоящим вирусам КВ прячутся, размножаются и ищут возможности перейти на другие ЭВМ.

Несмотря на широкую распространенность антивирусных программ, вирусы продолжают плодиться. В среднем в день появляется около 300 новых разновидностей.

Главная опасность - программы-вирусы начинают жить собственной жизнью, практически не зависящей от разработчика программы. Так же, как в цепной реакции в ядерном реакторе, запущенный процесс трудно остановить.

ИСТОРИЯ КОМПЬЮТЕРНЫХ ВИРУСОВ

Первые исследования саморазмножающихся искусственных конструкций проводилась в середине прошлого столетия учеными фон Нейманом и Винером.

Первый прототип вируса появился еще в 1971г . Программист Боб Томас, пытаясь решить задачу передачи информации с одного компьютера на другой, создал программу Creeper, самопроизвольно «перепрыгивавшую» с одной машины на другую в сети компьютерного центра. Правда эта программа не саморазмножалась, не наносила ущерба.

***Первая «эпидемия»** компьютерного вируса произошла в 1986 году, когда вирус по имени Brain (англ. «мозг») «заражал» дискеты персональных компьютеров. В настоящее время известно несколько десятков тысяч вирусов, заражающих компьютеры и распространяющихся по компьютерным сетям.*

Признаки вирусного заражения ЭВМ:

- *Замедление работы некоторых программ*
- *Увеличение размеров файлов (особенно выполняемых)*
- *Появление не существовавших ранее «странных» файлов*
- *Уменьшение объема доступной оперативной памяти (по сравнению с обычным режимом работы)*
- *Внезапно возникающие разнообразные видео и звуковые эффекты*
- *Появление сбоев в работе ОС (в т.ч. зависание)*
- *Запись информации на диски в моменты времени, когда этого не должно происходить*
- *Прекращение работы или неправильная работа ранее нормально функционировавших программ.*

Классификация вирусов.

По среде обитания:

***Сетевые** – распространяются по сетям (Melissa).*

***Файловые** – инфицируют исполняемые файлы с расширениями .exe, .com.*

Также к этому классу относятся макровирусы, которые заражают неисполняемые файлы (например, в MS WORD или в MS EXCEL).

***Загрузочные** – внедряются в загрузочный сектор диска* (Boot-сектор) или в сектор, содержащий программу загрузки системного диска (Master Boot Record - MBR). Некоторые вирусы записывают свое тело в свободные сектора диска, помечая их в FAT как «плохие».

***Файлово-загрузочные** – способны заражать и загрузочные секторы и файлы.*

По способу заражения:
Резидентные *– оставляют в оперативной памяти свою резидентную часть, которая затем перехватывает обращения программ к ОС и внедряется в них. Свои деструктивные действия вирус может повторять многократно.*

Нерезидентные *– не заражают оперативную память и проявляют свою активность лишь однократно при запуске зараженной программы.*

По степени опасности:

Неопасные *– например, на экране появляется сообщение: «Хочу чучу». Если набрать на клавиатуре слово «чуча», то вирус временно «успокаивается».*

Опасные *– уничтожают часть файлов на диске.*

Очень опасные *– самостоятельно форматируют жесткий диск. (CIH – активизируется 26 числа каждого месяца и способен уничтожать данные на жестком диске и в BIOS).*

По особенностям алгоритма:

Вирусы-компаньоны *– создают для exe-файлов новые файлы-спутники, имеющие то же имя, но с расширением com.*

Вирус записывается в com-файл и никак не изменяет одноименный exe-файл. При запуске такого файла ОС первым обнаружит и выполнит com-файл, т.е. вирус, который затем запустит и exe-файл.

Паразитические *– изменяют содержимое дисковых секторов или файлов.*

Репликаторы (черви) *– распространяются в сети. Они проникают в память компьютера из сети, вычисляют сетевые адреса других компьютеров и рассылают по этим адресам свои копии. Черви уменьшают пропускную способность сети, замедляют работу серверов.*

Могут размножаться без внедрения в другие программы и иметь «начинку» из компьютерных вирусов. («Червь Морриса» в конце 80-х парализовал несколько глобальных сетей в США).

Невидимки (стелс) *– маскируют свое присутствие в ЭВМ, их трудно обнаружить. Они перехватывают обращения ОС к пораженным файлам или секторам дисков и «подставляют» незараженные участки файлов.*

Мутанты (призраки, полиморфные вирусы, полиморфики) *– их трудно обнаружить. Это достигается тем, что в программы вирусов добавляются*

пустые команды (мусор), которые не изменяют алгоритм работы вируса, но затрудняют их выявление.

Макро-вирусы *– используют возможности макроязыков, встроенных в системы обработки данных (Word, Excel).*

«Троянские кони» *– маскируются под полезную или интересную программу, выполняя во время своего функционирования еще и разрушительную работу*
(например, стирает FAT) или собирает на компьютере информацию, не подлежащую разглашению. *Не обладают свойством самовоспроизводства.*

По целостности:

Монолитные *– программа вируса - единый блок, который можно обнаружить после инфицирования.*

Распределенные *– программа разделена на части. Эти части содержат инструкции, которые указывают компьютеру, как собрать их воедино, чтобы воссоздать вирус.*

Для борьбы с вирусами разрабатываются антивирусные программы. Говоря медицинским языком, эти программы могут выявлять (диагностировать), лечить (уничтожать) вирусы и делать прививку «здоровым» программам.

Виды антивирусных программ:

Программы-детекторы (сканеры) *– рассчитаны на обнаружение конкретных вирусов.*

Основаны на сравнении характерной (специфической) последовательности байтов, которые содержатся в теле вируса, с байтами проверяемых программ.

Эти программы нужно регулярно обновлять, т.к. они быстро устаревают и не могут выявлять новые виды вирусов. Если программа не опознается детектором как зараженная, это еще не значит, что она «здорова». В ней может быть вирус, который не занесен в базу данных детектора.

Программы-доктора (фаги, дезинфекторы) *– не только находят файлы, зараженные вирусом, но и лечат их, удаляя из файла тело программы-вируса.*
Полифаги – позволяют лечить большое число вирусов.

Широко распространены программы-детекторы, одновременно выполняющие и функции программ-докторов. Примеры: AVP (автор Е. Касперский), Aidstest (Д. Лозинский), Doctor Web (И. Данилов).

Программы-ревизоры *– анализируют текущее состояние файлов и системных областей дисков и сравнивают его с информацией, сохраненной ранее в одном из файлов ревизора.*

При этом проверяется состояние Boot-сектора, FAT, а также длина файлов, их время создания, атрибуты, контрольные суммы (суммирование по модулю 2 всех байтов файла).

Пример такой программы – Adinf (Д. Мостовой).

Программы-фильтры *(сторожа) – резидентные программы, которые оповещают пользователя обо всех попытках какой-либо программы выполнить подозрительные действия, а пользователь принимает решение о разрешении или запрещении выполнения этих действий.*
Фильтры контролируют следующие операции: обновление программных файлов и системной области дисков; форматирование диска; резидентное размещение программ в ОЗУ. Примером служит программа Vsafe. Она не способна обезвредить вирус, для этого нужно использовать фаги.

Программы-иммунизаторы *– записывают в вакцинируемую программу признаки конкретного вируса так, что вирус считает ее уже зараженной, и поэтому не производит повторное инфицирование.*

Эти программы наименее эффективны и морально устарели.

Меры по защите ЭВМ от заражения вирусами:

- *Оснащение ЭВМ современными антивирусными программами и регулярное обновление их версий.*
- *Установка программы-фильтра при работе в глобальной сети.*
- *Проверка внешних носителей информации на наличие вирусов перед считыванием информации, записанной на других ЭВМ.*
- *При переносе на свой ПК файлов в архивированном виде проверка их сразу после разархивации.*

- *Создание архивных копий ценной информации на других носителях информации.*

Не оставлять дискету в дисководе при включении или перезагрузки ПК, т.к. возможно заражение загрузочными вирусами. Наличие аварийной загрузочной дискеты, с которой можно будет загрузиться, если система откажется сделать это обычным образом.

АРХИВАЦИЯ ДАННЫХ

*Характерной особенностью большинства типов данных является **их избыточность**. Степень избыточности данных зависит от типа данных. Например, для видеоданных степень избыточности в несколько раз больше чем для графических данных, а степень избыточности графических данных, в свою очередь, больше чем степень избыточности текстовых данных. Другим фактором, влияющим на степень избыточности, является принятая система кодирования. Примером систем кодирования могут быть обычные языки общения, которые являются ни чем другим, как системами кодирования понятий и идей для высказывания мыслей. Так, установлено, что кодирование текстовых данных с помощью средств русского языка дает в среднем избыточность на 20-25% большую, чем кодирование аналогичных данных средствами английского языка.*

Для человека избыточность данных часто связана с качеством информации, поскольку избыточность, как правило, улучшает понятность и восприятие информации. Однако, когда речь идет о хранении и передаче информации средствами компьютерной техники, то избыточность играет отрицательную роль, поскольку она приводит к возрастанию стоимости хранения и передачи информации. Особенно актуальной эта проблема встает в случае обработки огромных объемов информации при незначительных объемах носителей данных. В связи с этим, постоянно возникает проблема уменьшения избыточности или сжатия данных.

Сжатие данных – это архивация данных.

*Сжатый (упакованный) файл называется **архивом.***

*Процесс записи файла в архивный файл называется **архивированием** (упаковкой, сжатием), а извлечение файла из архива – **разархивированием** (распаковкой).*

Архивация информации *– это такое преобразование информации, при котором объем информации уменьшается, а количество информации остается прежним.*

В зависимости от того, в каком объекте размещены данные, подлежащие сжатию, различают:

1. ***Сжатие (архивация) файлов****: используется для уменьшения размеров файлов при подготовке их к передаче каналами связи или к транспортированию на внешних носителях маленькой емкости;*
2. ***Сжатие (архивация) папок****: используется как средство уменьшения объема папок перед долгим хранением, например, при резервном копировании;*
3. ***Сжатие (уплотнение) дисков****: используется для повышения эффективности использования дискового пространства путем сжатия данных при записи их на носителе информации (как правило, средствами операционной системы).*

Способы уменьшения избыточности данных:

1. *Изменение содержимого данных.*
2. *Изменение структуры данных.*
3. *Одновременное изменение, как структуры, так и содержимого данных.*

Методы сжатия:

1. ***Необратим****ые (методы сжатия с регулированными потерями информации). Эти методы применимы к типу данных, для которых потеря данных не приводит к существенному искажению информации. К ним относятся форматы:*
    - *JPEG - для графических данных;*
    - *MPG - для видеоданных;*
    - *MP3 - для аудиоданных.*
2. ***Обратимые*** *(методы сжатия без потерь информации). Происходит только изменение структуры данных. Из архива можно восстановить информацию полностью.*

Примеры форматов:

- *GIF, TIFF - для графических данных;*

- *AVI - для видеоданных;*
- *ZIP, ARJ, RAR, CAB, LH - для произвольных типов данных.*

*В основе **методов сжатия** информации лежат три теоретических алгоритма:*

- *алгоритм RLE (Run Length Encoding);*
- *алгоритм Лемпеля-Зива;*
- *алгоритм Хаффмана.*

Алгоритм RLE

В основе алгоритма RLE лежит идея выявления повторяющихся последовательностей данных и замены их более простой структурой, в которой указывается код данных и коэффициент повторения.

Например, пусть задана такая последовательность данных, что подлежит сжатию:

1 1 1 1 2 2 3 4 4 4

*В алгоритме RLE предлагается заменить ее следующей структурой: 1 4 2 2 3 1 4 3, где **первое число** каждой пары чисел - **это код данных**, а **второе - коэффициент повторения.***

Если для хранения каждого элемента данных входной последовательности отводится 1 байт, то вся последовательность будет занимать 10 байт памяти, тогда как выходная последовательность (сжатый вариант) будет занимать 8 байт памяти.

Алгоритм Лемпеля-Зива

Популярные архиваторы ARJ, PAK, PKZIP работают на основе алгоритма
Лемпела-Зива. Эти архиваторы классифицируются как адаптивные словарные кодировщики, в которых текстовые строки заменяются указателями на идентичные им строки, встречающиеся ранее в тексте.

Например, все слова какой-нибудь книги могут быть представлены в виде номеров страниц и номеров строк некоторого словаря. Важнейшей отличительной чертой этого алгоритма является использование грамматического разбора предшествующего текста с расположением его на фразы, которые записываются в словарь. Указатели позволяют сделать ссылки на любую фразу в окне установленного размера, предшествующего текущей фразе.

Алгоритмы сжатия этой группы наиболее эффективны для текстовых данных больших объемов и малоэффективны для файлов маленьких размеров (за счет необходимости сохранение словаря).

Алгоритм Хаффмана

В основе алгоритма Хаффмана лежит идея кодирования битовыми группами. Сначала проводится частотный анализ входной последовательности данных, то есть устанавливается частота вхождения каждого символа, встречающегося в ней. После этого, символы сортируются по уменьшению частоты вхождения.

Основная идея состоит в следующем: чем чаще встречается символ, тем меньшим количеством бит он кодируется. Результат кодирования заносится в словарь, необходимый для декодирования. Рассмотрим простой пример, иллюстрирующий работу алгоритма Хаффмана.

Пусть задан текст, в котором бурва 'A' входит 10 раз, буква 'B' - 8 раз, 'C'- 6 раз , 'D' - 5 раз, 'E' и 'F' - по 4 раза. Тогда один из возможных вариантов кодирования по алгоритму Хаффмана приведен в таблицы 1.

Таблица 1.

Символ	*Частота вхождения*	*Битовый код*
A	*10*	*00*
B	*8*	*01*
C	*6*	*100*
D	*5*	*101*
E	*4*	*110*
F	*4*	*111*

Как видно из таблицы 1, размер входного текста до сжатия равен 37 байт, тогда как после сжатия - около 12 байт (без учета длины словаря).

Коэффициент сжатия равен 32%.

Алгоритм Хаффмана универсальный, его можно применять для сжатия данных любых типов, но он малоэффективен для файлов маленьких размеров (за счет необходимости сохранение словаря).

Языки программирования

Программирование *- это искусство создавать программные продукты, которые написаны на языке программирования.*

Язык программирования *– это формальная знаковая система, которая предназначена для написания программ, понятной для исполнителя (в нашем рассмотрении – это компьютер).*

Язык программирования *– система обозначений для описания алгоритмов и структур данных, определенная искусственная формальная система, средствами которой можно выражать алгоритмы.*

Язык программирования *определяет набор лексических, синтаксических и семантических правил, задающих внешний вид программы и действия, которые выполняет исполнитель (компьютер) под ее управлением.*

Со времени создания первых программируемых машин было создано более двух с половиной тысяч языков программирования. Ежегодно их число пополняется новыми. Некоторыми языками умеет пользоваться только небольшое число их собственных разработчиков, другие становятся известны миллионам людей. Профессиональные программисты обычно применяют в своей работе несколько языков программирования.

Языки программирования низкого уровня

Первым компьютерам приходилось программировать двоичными машинными кодами. Однако программировать таким образом - достаточно трудоемкая и сложная задача. Для упрощения этой задачи стали появляться ***языки программирования*** *низкого уровня, которые позволяли задавать машинные команды в более понятном для человека виде. Для преобразования их в двоичный код были созданы специальные* ***программы - трансляторы****.*

Трансляторы делятся на:

- ***компиляторы*** *- превращают текст программы в машинный код, который можно сохранить и затем использовать уже без компилятора (примером являются исполняемые файлы с расширением *. exe).*
- ***интерпретаторы*** *- превращают часть программы в машинный код, выполняют и после этого переходят к следующей части. При этом каждый раз при выполнении программы используется интерпретатор.*

Примером языка низкого уровня является ***ассемблер,*** *который применяется до сих пор.*

Языки низкого уровня ориентированы на конкретный тип процессора и учитывают его особенности, поэтому для переноса программы на ассемблере на другую аппаратную платформу ее нужно почти полностью переписать.

Преимущества

С помощью языков низкого уровня создаются эффективные и компактные программы, поскольку разработчик получает доступ ко всем возможностям процессора.

Недостатки

- *Программист, работающий с языками низкого уровня, должен быть высокой квалификации, хорошо понимать устройство микропроцессорной системы, для которой создается программа.*
- *результирующая программа не может быть перенесена на компьютер или устройство с другим типом процессора.*
- *значительное время разработки больших и сложных программ.*

Языки низкого уровня, как правило, используют для написания небольших системных программ, драйверов устройств, модулей стыков с нестандартным оборудованием, программирование специализированных микропроцессоров, когда важнейшими требованиями являются компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам.

Языки программирования высокого уровня

К ним относятся:

- *Фортран*
- *Кобол*
- *Алгол*
- *Pascal*
- *Java*
- *C*
- *C++*
- *C#*
- *Objective C*
- *Smalltalk*
- *Delphi*

Преимущества.

- *Особенности конкретных компьютерных архитектур в них не учитываются, поэтому созданные программы легко переносятся с компьютера на компьютер.*

- *Достаточно просто перекомпилировать программу под определенную компьютерную архитектурную и операционную систему.*

- *Разрабатывать программы на таких языках гораздо проще и ошибок допускается меньше.*

- *Значительно сокращается время разработки программы*

Недостатки.

Больший размер программ по сравнению с программами на языке низкого уровня.

Поэтому в основном языки высокого уровня используются для разработок программного обеспечения компьютеров и устройств, которые имеют большой объем памяти. А разные подвиды ассемблера применяются для программирования других устройств, где критичным является размер программы.