Конспект для подготовки к экзамену по предмету «Информатика».

Преподаватель: **Ищенко Алексей Петрович**

1й семестр, факультет ИТиП, кафедра ИС

Подготовили и оформили: Трофимов Владислав, Махонин Кирилл (вторая версия)

Предложения и ошибки просьба отправлять на 172432@niuitmo.ru, stranger.65536@gmail.com

Конспект для подготовки к экзамену по предмету основы программирования ориентировочно будет готов к второй-третьей недели января.

Конспекты по математике ожидаются к 9 января.

Конспекты по физике не гарантируются.

СОДЕРЖАНИЕ

Информация, данные, единицы измерения информации, формализация и моделирование.

Предметом изучения информатики являются данные: методы их создания, хранения, обработки и передачи. *Информация* - то, что зафиксировано в данных.

Наименьшая единица измерения информации - *бит*. Может принимать одно из двух состояний - 0 или 1. Последовательность из 8 битов называется байтом. 1Кб = 1024Б, 1Мб = 1024Кб и т.д. Последовательностью из N байт можно закодировать 2^{8*N} различных состояний.

Информационная модель – упрощенное подобие реального объекта, который отражает существенные особенности (свойства) изучаемого реального объекта, явления или процесса.

Моделирование – метод познания, состоящий в создании и исследовании моделей. Т.е. исследование объектов путем построения и изучения моделей.

Формализация – процесс построения информационных моделей с помощью формальных языков.

Системы счисления, позиционные, непозиционные. Что такое число, цифра.

Число - описание количества при помощи специальных знаков и правил. *Цифры* - система знаков для обозначения чисел. Система счисления - набор знаков и правил для обозначения чисел.

Позиционные системы счисления - каждая цифра находится на определенной позиции и имеет свой вес. *Основание системы счисления* - коэффициент, показывающий, во сколько раз единица одного разряда отличается от единицы соседнего разряда.

Непозиционные системы счисления - для построения чисел применяются другие правила (Римская система счисления).

1986_{10} = 1000 + (1000 - 100) + 50 + 10 + 10 + 10 + 5 + 1 = MCMLXXXVI

M = 1000. D = 500. C = 100. L = 50. X = 10. V = 5. I = 1. Если большая цифра стоит перед меньшей, то они складываются. Если меньшая перед большей, то из большей вычитается меньшая (Применяется во избежание повторения более трех одинаковых цифр подряд.)

Правила перевода в десятичную систему счисления и обратно.

При переводе числа из любой системы счисления в десятичную необходимо расставить над каждой цифрой числа разряд, начиная с наименьшего целого разряда (он обозначается нулем).

Далее, все разряды левее обозначаются с увеличением на единицу, правее - с уменьшением на единицу

Пример:

$(5_4 6_3 7_2 8_1 9_0)_{16}$

Далее, для каждого разряда, начиная с самого левого, умножаем значение разряда на основание системы счисления в степени номера данного разряда и складываем все полученные числа .

Пример:

$(5_4 6_3 7_2 8_1 9_0)_{16} = 5*16^4 + 6*16^3 + 7*16^2 + 8*16^1 + 9*16^0 = 480905$

При переводе числа из десятичной системы в любую, целая и дробная части переводятся отдельно. Для перевода целой части числа необходимо в столбик разделить число на основание системы счисления, потом частное опять разделить на основание системы счисления, и так до тех пор, пока частное не станет меньше основания системы счисления.

Результатом перевода будет последовательность последнего частного и остатков от деления, начиная с последнего и заканчивая первым.

Пример:

Перевести 15_{10} в двоичную систему счисления

```
15 / 2 = 7 * 2 + 1
7 / 2 = 3 * 2 + 1
3 / 2 = 1 * 2 + 1.
x = 1111₂.
```

Для перевода дробной части числа необходимо умножать дробную часть на основание системы счисления, записывая на каждом шаге целую часть от умножения, и принимать на следующем шаге умножения целую часть равной нулю до тех пор, пока не будет достигнута необходимая точность, либо пока дробная часть не станет равной нулю.

Пример:

Перевести 0.65625_{10} в восьмеричную систему счисления

```
0.65625 * 8 = 5.25
0.25 * 8 = 2.00
дробная часть = 0, расчет окончен, x = 0.52₈
```

Ответ: 0.52_8

Системы счисления с основаниями, кратными двум.

Для систем счисления, с основанием кратных двум, при переводе из одной такой системы счисления в другую, при условии что основание одной такой системы можно получить из другой возведением в степень (например 2 и 16, 8 и 64, 16 и 4, 2 и 8, 4 и 16), можно применять *правило ускоренного перевода*.

Для этого надо вычислить N = log(большая С.С.) / log(меньшая С.С.)

При переводе из большей системы счисления в меньшую, заменить каждую цифру на ее представление в меньшей системе счисления ровно N цифрами, при необходимости добавляя незначащие нули слева.

Пример

Перевести $F17_{16}$ в четверичную систему счисления

N = 2;
$F_{16} = 33_4$;
$1_{16} = 01_4$;
$7_{16} = 13_4$;
Ответ: 330113_4

При переводе из меньшей в большую, начиная от наименьшего целого разряда, заменить группы по N чисел на их представление в требуемой системе счисления, добавляя при необходимости незначащие нули слева.

Пример

Перевести 30113_4 в шестнадцатеричную систему счисления

N = 2;

$30113_4 = 030113_4$; (Дополнили слева незначащими нулями)

$03_4 = 3_{16}$;

$01_4 = 1_{16}$;

$13_4 = 7_{16}$;

Ответ: 317_{16}.

Метод ускоренного перевода можно применять только для целой части числа.

Дополнительный код двоичного числа, отрицательные двоичные числа.

Дополнительный код двоичного числа получается путем применения к каждому биту числа, кроме значащего (старшего), операцию отрицания (0 -> 1; 1 -> 0) и прибавлением единицы.

Так же можно использовать вычитание данного числа из нуля.

Дополнительный код позволяет заменить операцию вычитания на операцию сложения и тем самым сделать операции сложения и вычитания одинаковыми для знаковых и беззнаковых чисел. Для знакового типа старший бит является знаковым (0 – положительное число, 1 - отрицательное).

Для положительных знаковых и беззнаковых чисел:

В двоичном представлении прямой, обратный и дополнительный коды совпадают.

Для отрицательных:

Старший бит - знаковый, остальные биты инвертируются, и к самому младшему биту прибавляется единица.

Пример:

127_{10}: Прямой код = 01111111_2 = Обратный = Дополнительный.

-127_{10}: Прямой код = 11111111_2, Обратный = 10000000_2, Дополнительный = 10000001_2.

Нотация с избытком, дробные двоичные числа, порядок, мантисса, знак.

Для дробных двоичных чисел можно применять *однобайтное представление*: [знак][порядок][мантисса]. Под знак отводится 1 бит, под порядок 3, под мантиссу 4.

Знак

- 0 – Число положительно
- 1 – Число отрицательно

Порядок показывает, насколько нужно сдвинуть точку относительно мантиссы. Порядок записан в виде нотации с избытком.

Мантисса – значащие цифры числа в прямом представлении.

В двухбайтном представлении под порядок отводится 4 бита, под мантиссу 11.

Пример:

`1 0101 01101101101`

- -3 0.000***01101101101***

Ответ: $-\frac{877}{16384}$

Если порядок положителен, то мантисса должна начинаться с единицы. Если отрицателен, то заканчиваться на единицу.

Двоичная нотация с избытком

Для любой системы счисления вычитаем из числа `(основание системы счисления)^(максимальная длина числа – 1)`. **Пример**:

001_2 = 1. 1 – 4 = -3. (вычитаем 4 т.к. $2^{(3-1)}$=4)

Число при нотации с избытком	«Реальное» значение числа
111	3
110	2
101	1
100	0
011	-1
010	-2
001	-3
000	-4

Арифметические действия с двоичными числами, сдвиг влево, сдвиг вправо

Арифметические действия:

- Сложение
- Вычитание (за счет дополнительного кода сводится к операции сложения)
- Сдвиг влево
- Сдвиг вправо

Сдвиг влево аналогичен операции умножить на 2. Сдвиг вправо аналогичен операции разделить на 2 без остатка.

При выходе числа за границы типа данных, все, что не поместилось, отбрасывается.

Любой новый байт, который появляется при сдвиге, равен 0.

Число	0	0	0	0	**1**	**0**	**1**	**0**
Число << 2	0	0	**1**	**0**	**1**	**0**	0	0

Число	0	0	0	0	**1**	**1**	**1**	**0**
Число >> 3	0	0	0	0	0	0	0	**1**

Логические операции и их свойства

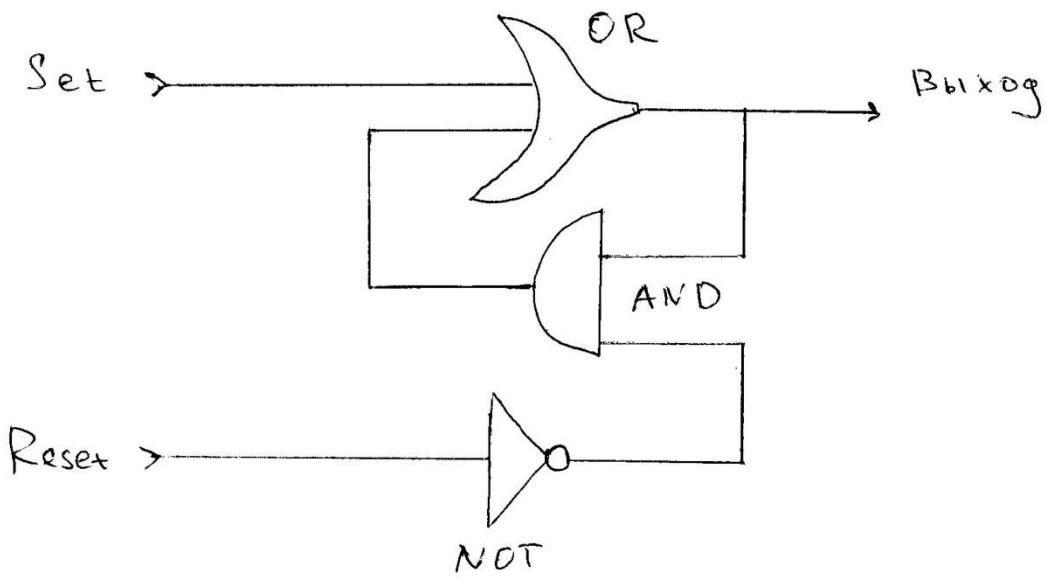
Операция	Описание	a	b	Результат
! a, NOT a, $\overline{a}$	Отрицание. Инвертирует биты (1=0, 0=1)	1	-	0
$a * b$, $a \wedge b$	Логическое И (1 и 1 = 1, иначе 0)	1	0	0
$a + b$, $a \vee b$	Логическое ИЛИ (0 и 0 = 0, иначе 1)	0	1	1
$a\ XOR\ b$, $a \oplus b$	XOR (0 и 0 = 0, 1 и 1 = 0, иначе 1)	1	0	1
$a <-> b$, $a \sim b$	Эквивалентность, равенство. (0 и 0 = 1, 1 и 1 = 1, иначе 0)	1	1	1
$a \rightarrow b$	Логическое следование (1 и 0 = 0, иначе 1)	1	1	1

Приоритет операций:

1. not
2. and
3. or,$\oplus$
4. $\rightarrow$
5. $\sim$

Триггеры, схемы простейших триггеров

Триггер (триггерная система) — класс электронных устройств, обладающих способностью длительно находиться в одном из двух устойчивых состояний и чередовать их под воздействием внешних сигналов.



По умолчанию, **Set** и **Reset** не имеют сигнала.

При подаче сигнала (даже кратковременной) на вход **Set** внутри триггерной системы (путь OR-AND-OR-выход) создается сигнал на **Выход**. Так как сигнал на **Reset** не поступает, то путь Not-And имеет сигнал, следовательно, сигнал на пути OR-AND курсирует без изменений.

При подаче сигнала на **Reset**, путь Not-And теряет сигнал, из за этого внутри триггерной системы сигнал обнуляется (если он присутствовал), на **Выход** сигнал не подается.

Законы алгебры логики.

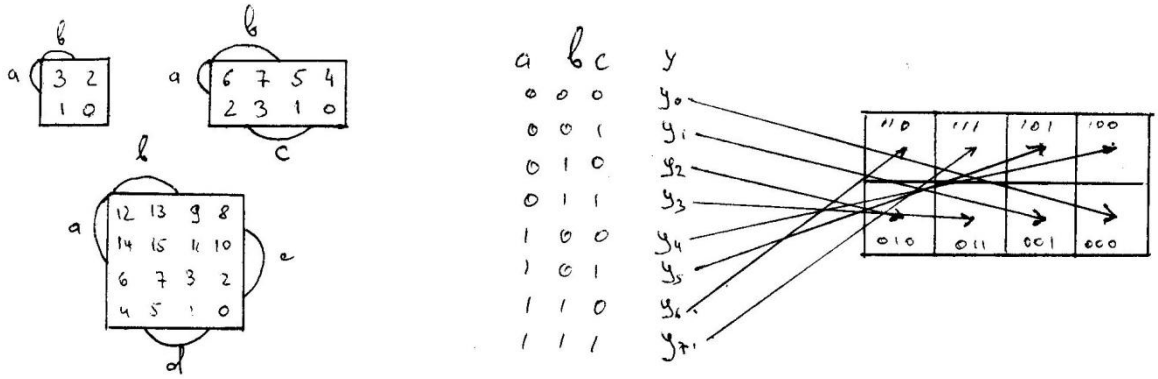
- Коммутативность: x$\circ$y = y$\circ$x, $\circ \in$ {&, $\vee, \oplus, \sim, |, \downarrow$}.
- Идемпотентность: x$\circ$x = x, $\circ \in$ {&, $\vee$}.
- Ассоциативность: (x$\circ$y)$\circ$z = x$\circ$(y$\circ$z), $\circ \in$ {&, $\vee, \oplus, \sim$}.
- Дистрибутивность конъюнкций и дизъюнкции относительно дизъюнкции, конъюнкции и суммы по модулю два соответственно:
  - $x \wedge (y \vee z) = x \wedge y \vee x \wedge z$,
  - $x \vee y \wedge z = (x \vee y) \wedge (x \vee z)$,
  - $x \wedge (y \oplus z) = x \wedge y \oplus x \wedge z$.
- Законы де Моргана:
  - $\overline{x \wedge y} = \bar{x} \vee \bar{y}$,
  - $\overline{x \vee y} = \bar{x} \wedge \bar{y}$.
- Законы поглощения:
  - $x \wedge (x \vee y) = x$,
  - $x \vee x \wedge y = x$.
- Другие:
  - $x \wedge \bar{x} = x \wedge 0 = x \oplus x = 0$.
  - $x \vee \bar{x} = x \vee 1 = x \sim x = x \to x = 1$.
  - $x \vee x = x \wedge x = x \wedge 1 = x \vee 0 = x \oplus 0 = x$.
  - $x \oplus 1 = x \to 0 = x \sim 0 = x \mid x = x \downarrow x = \bar{x}$.
  - $\bar{\bar{x}} = x$, инволютивность отрицания, закон снятия двойного отрицания.
  - $x \oplus y = x \wedge \bar{y} \vee \bar{x} \wedge y = (x \vee y) \wedge (\bar{x} \vee \bar{y})$.
  - $x \sim y = \overline{x \oplus y} = 1 \oplus x \oplus y = x \wedge y \vee \bar{x} \wedge \bar{y} = (x \vee \bar{y}) \wedge (\bar{x} \vee y)$.
  - $x \to y = \bar{x} \vee y = x \wedge y \oplus x \oplus 1$.
  - $x \vee y = x \oplus y \oplus x \wedge y$.
  - $x \mid y = \overline{x \wedge y} = \bar{x} \vee \bar{y}$.
  - $x \downarrow y = \overline{x \vee y} = \bar{x} \wedge \bar{y}$.

Представление таблицы истинности в виде карты Карно, упрощение логических выражений.

Карты Карно позволяют получить логическое выражение для таблицы истинности.

Рассмотрим карты Карно для таблиц истинности двух, трех и четырех переменных.

Каждой ячейки в карте соответствует число, показывающее на номер строки в таблице истинности (начиная с нулевой строки).

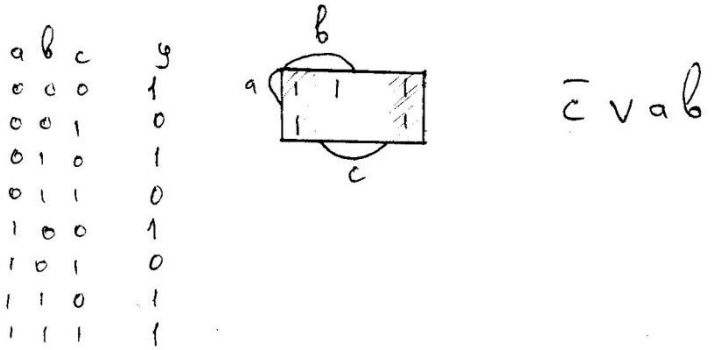


Карты Карно можно заполнять не используя номера, а используя маркировку столбцов (a,b,c,d). Если ячейка расположена в столбце/строке с маркировкой (например a), то маркер является истинным.

Пример

Получить логическое выражение для таблицы истинности.

1. Чертим Карту Карно на 3 переменные
2. Заполняем карту Карно
3. Выписываем функцию путем учета позиций *1*. (В примере все *1* расположены вне столбца *c* или в пересечении *a* и *b*).



Код Грея, схемы и правила преобразования двоичного числа.

Код Грея используется в счетчиках для последовательного наращивания методом изменения только одного бита. Число в коде Грея отличается от предыдущего только на один разряд. Изменяется минимальный разряд на противоположный.

Преобразование в код Грея

Коды Грея легко получаются из двоичных чисел путём побитовой операции «Исключающее ИЛИ» с тем же числом, сдвинутым вправо на один бит. Следовательно, *i*-й бит кода Грея G_i выражается через биты двоичного кода B_i следующим образом:

$$G_i = B_i \oplus B_{i+1},$$

Цифра, стоящая на позиции i в коде Грея равна результату XOR цифр, стоящих на позициях i и $i+1$ в двоичном числе.

Двоичн.	Грей
000	000
001	001
010	011
011	010
100	110
101	111
110	101
111	100

Пример:

Преобразовать 10110 в код Грея

```
10110
01011
-----
11101
```

Перевод из кода Грея

Первую цифру переносим. Каждая последующая получается XOR предыдущей полученной и цифрой, стоящей на той же позиции в коде Грея, что и искомая

Пример:

Преобразовать код Грея 11101 в двоичный код

```
11101
1xxxx
10xxx 0 = 1 XOR 1
101xx 1 = 0 XOR 1
1011x 1 = 1 XOR 0
10110 0 = 1 XOR 1
-----
10110
```

Операционная система и ее свойства

В компьютере есть две составляющие – hardware и software.

Software подразделяется на 4 уровня:

1. Base-level (firmware).
2. System-level.
3. Service-level.
4. User-level.

Операционная система – комплекс программных средств, обеспечивающих работу и взаимодействие пользователя с программами, программ с оборудованием, организующий хранение и доступ к данным, обмен данными между программами и устройствами.

Основные функции: менеджер ресурсов, защита данных пользователя и программ, обеспечение работы всех программ и организация доступа к ресурсам и взаимодействие, постоянно работающее ядро, виртуализация ресурсов.

Основные параметры ОС и их характеристики:

- Разрядность показывает скольки разрядными командам работает ОС (16,32,64)
- Платформа с точки зрения применимости ПК – стационарная, мобильная, серверная
- Интерфейс – средство взаимодействия с устройством – CLI, GUI, SILK
- Многозадачность – однозадачность (задачи выполняются последовательно одна за другой), псевдомногозадачность (запуск множества задач, но в определенный момент выполняется только одна и них), многозадачность (одновременное выполнение нескольких задач на разных ядрах)
- Файловая система – родная или поддерживаемая
- Многопользовательность – однопользовательская однозадачная, однопользовательская многозадачная, многозадачная многопользовательская, однопользовательская псевдомногозадачная
- Сетевые возможности — клиент (потребитель внешних ресурсов), сервер (поставляет ресурсы)
- Организация и распределение памяти между процессором и ОС
- Поддержка многопроцессорности
- Архитектура ядра — микроядерность, монолитность ядра, слоеные системы (каждый уровень – отдельная оболочка), виртуальные машины, гибридные ядра.
- Требования к системе (конфигурация ПК) – минимальные и оптимальные
- Надежность ОС – стабильность работы и механизмы защиты от сбоев
- Безопасность – надежность данных пользователя и невозможность повреждения их другими пользователями. **Это все есть в orange book – стандарт безопасности ОС A B C D**
- Интерактивность – ведение диалога с пользователем
- Открытость (модули) – возможность изменения конфигурации
- Программные и аппаратные утилиты, встроенные в систему
- История и предпосылки создания
- Версии ОС
- Мнение пользователей

Разновидности интерфейсов и их применение.

- *CLI* – command line interface – обработка командным интерпретатором. Применяется в основном в серверных системах, где наличие GUI существенно замедляет общую работу системы.
- *GUI* – graphic user interface – наличие пассивных (окно, меню, иконка) и активных (курсор) графических элементов.
- *SILK* – speech image language knowledge – обучаемый распознаватель образов. (Siri, голосовой поиск Google)

Командный интерфейс операционной системы, основные команды для работы с файлами.

CD Вывод имени либо смена текущей папки.
CHDIR Вывод имени либо смена текущей папки.
CHKDSK Проверка диска и вывод статистики.
CLS Очистка экрана.
CMD Запуск еще одного интерпретатора командных строк Windows.
COMP Сравнение содержимого двух файлов или двух наборов файлов.
COPY Копирование одного или нескольких файлов в другое место.
DEL Удаление одного или нескольких файлов.
DIR Вывод списка файлов и подпапок из указанной папки.
DISKCOMP Сравнение содержимого двух гибких дисков.
DISKCOPY Копирование содержимого одного гибкого диска на другой.
DISKPART Отображение и настройка свойств раздела диска.
ECHO Вывод сообщений и переключение режима отображения команд на экране.
ERASE Удаление одного или нескольких файлов.
EXIT Завершение работы программы CMD.EXE (интерпретатора командных строк).
FIND Поиск текстовой строки в одном или нескольких файлах.
FINDSTR Поиск строк в файлах.
FOR Запуск указанной команды для каждого из файлов в наборе.
FSUTIL Отображение и настройка свойств файловой системы.
FTYPE Вывод либо изменение типов файлов, используемых при сопоставлении по расширениям имен файлов.
IF Оператор условного выполнения команд в пакетном файле.
MD Создание папки.
MKDIR Создание папки.
MOVE Перемещение одного или нескольких файлов из одной папкив другую.
OPENFILES Отображение файлов, открытых на общей папке удаленным пользователем.
PROMPT Изменяет приглашение в командной строке Windows.
PUSHD Сохраняет значение активной папки и переходит к другой папке.
RD Удаляет папку.
REN Переименовывает файлы или папки.
RENAME Переименовывает файлы или папки.
REPLACE Замещает файлы.
RMDIR Удаление папки.
ROBOCOPY Улучшенное средство копирования файлов и деревьев каталогов
SET Показывает, устанавливает и удаляет переменные среды Windows.
SHUTDOWN Локальное или удаленное выключение компьютера.
SUBST Назначение заданному пути имени диска.
TASKKILL Прекращение или остановка процесса или приложения.
TIME Вывод и установка системного времени.
TITLE Назначение заголовка окна для текущего сеанса интерпретатора командных строк CMD.EXE.
TREE Графическое отображение структуры каталогов диска или папки.
TYPE Вывод на экран содержимого текстовых файлов.
XCOPY Копирование файлов и деревьев каталогов.

Командный интерфейс операционной системы, основные команды для работы с сетью

IPCONFIG Вывод деталей текущего соединения и управление клиентскими сервисами DHCP и DNS
GETMAC Отобразить MAC адреса сетевых адаптеров
NBTSTAT Отображение статистики протокола и текущих подключений TCP/IP с помощью NetBIOS
NETSH Конфигурирование сетевых параметров
NETSTAT Отображение протокола и текущих сетевых подклчений
NSLOOKUP Обращение к системе DNS
PING Утилита для проверки соединения в сетях на основе TCP/IP
ROUTE Обработка таблиц сетевых маршрутов
TELNET Клиент для протокола Telnet
TRACERT Определение маршрута следования данных в сетях TCP/IP

Интерфейс командных оболочек на примере FAR Manager.

Программа *FAR Manager* наследует двухоконную идеологию, стандартную (по умолчанию) расцветку и систему команд (управление с клавиатуры) у известного файлового менеджера Norton Commander (рисунок 4).

Программа может работать как в оконном, так и полноэкранном режиме. Некоторые недостатки интерфейса, схожие с DOS-программами в оконном режиме (невозможность произвольного изменения текстового разрешения окон, проблемы с закрытием при выключении системы), проявляются в Windows 9x и отсутствуют в семействе Windows NT.

FAR поддерживает длинные имена файлов, атрибуты файлов файловой системы NTFS, различные кодировки текстов, может использовать системные функции для копирования файлов, имеет многоязычный интерфейс и систему помощи.

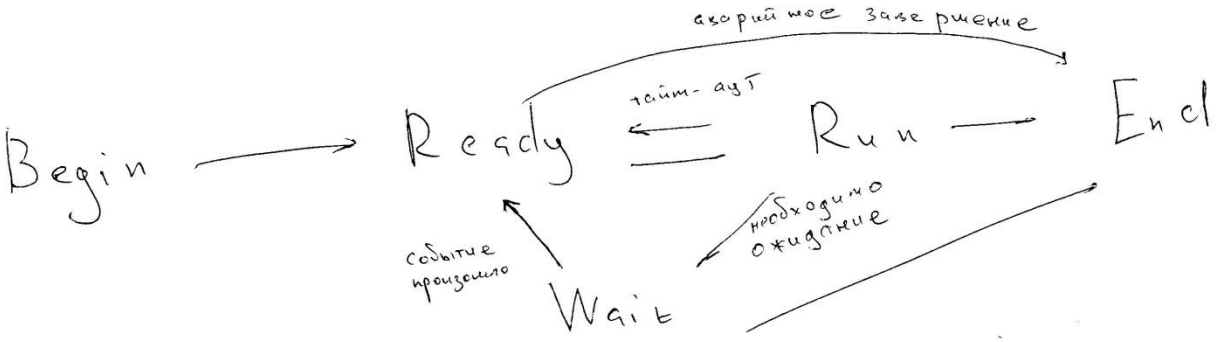
Понятие процесса, возможные состояния процесса.

Процесс – некоторая деятельность, связанная с исполнением программы на процессоре.

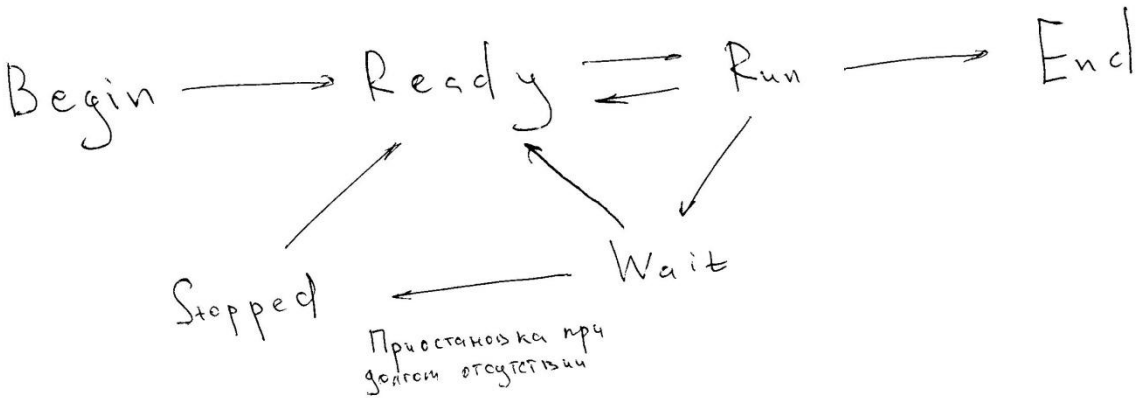
Состояния процесса:

1. **Порождение** – подготовка условий для первого исполнения на процессоре.
2. **Активное состояние** – программа исполняется на процессоре.
3. **Ожидание** – Программа не выполняется на процессоре в силу занятости какого-либо необходимого ресурса.
4. **Готовность** – программа не выполняется, но исполнения доступны все ресурсы, кроме процессора.
5. **Окончание** – нормальное/аварийное завершение исполнения программы, после которого ей не предоставляются никакие ресурсы.

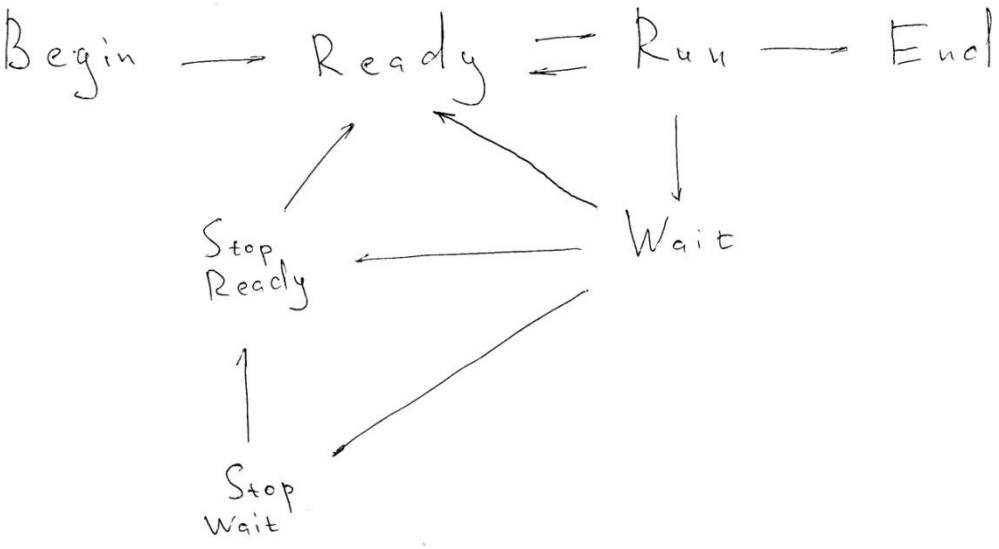
Пятиуровневая модель системы состояния процесса



Шестиуровневая модель системы состояния процесса



Семиуровневая модель системы состояния процесса



Классификация процессов

У процессов есть временные характеристики:

- *Интервал существования процесса* – промежуток времени от запуска до завершения процесса.
- *Интерактивные процессы* (зависимость времени нахождения от действий юзера) и пакетные (не завис.)
- *Порождающие* и *порожденные* процессы (родители и дети)

2 процесса, имеющие один и тот же конечный результат, использующие при обработке одни и те же исходные данные, одной той же или даже для различных программ на одном и том же или даже на различных процессорах, называются *эквивалентными*.

Трасса процесса – порядок и длительность процесса в допустимых состояниях на интервале существования. В общих случаях у эквивалентных процессов трассы не совпадают. Если два эквивалентных процесса исполняются по одной и той же программе, их называют *тождественными*. Если при этом у них совпадают трассы – *равными,* в ином случае они являются *неравными*. Если процессы зависят друг от друга – *последовательные*. Если нет – *параллельные*. Если процессы пересекаются – *комбинированные*.

По месту пребывания процессы делятся на:

- *внешние* (выполняются в системе пользователем и другими процессорами, отличными от центрального)
- *внутренние* (выполняются на ЦП)

По отношению к системе и пользователя

- *пользовательские*
- *системные*

По отношению друг к другу:

- *информационно-независимые*
- *взаимосвязанные*
- *изолированные*
- *конкурирующие*

Механизмы взаимодействия процессов и обмена информации между процессами

Особенности кооперации процессов: повышение скорости работы, необходимость разделения данных, удобство пользователя.

Механизмы взаимодействия процессов:

1. *Сигнальные* – минимальное количество передаваемой информации, сигнал другим процессам о состоянии текущего и занятости ресурсов. Почти не влияет на работу других процессов.
2. *Канальные* – общение через линии связи ОС. Объем информации ограничен пропускной способностью линии связи. Больше информации -> больше влияние на работу других процессов.
3. *Разделяемая память* – область адресного пространства, выделяемая ОС для совместного использования несколькими процессами. 3 и 2 инициализируются и регулируются с помощью ОС.

Способы адресации при установлении связи:

- *прямая* – явное указание на получателя и отправителя информации (симметричная адресация) либо явное указание только на получателя или отправителя (ассиметричная адресация)
- *непрямая* – не указывается отправитель и получатель.

Ни один процесс не может вмешаться в процедуру общения двух процессов с прямой симметричной адресацией.

Может потребоваться инициализация средства связи. Информация, необходимая процессу для обмена информацией – некий идентификатор промежуточного объекта хранения данных.

Валентность связи – сколько процессов одновременно инициализировано с этим средством связи.

Валентность процесса – Сколько средств связи инициализировано с процессом.

Направленность связи:

- *однонаправленная* – процесс выполняет передачу
- *параллельная* – с получением данных (симплексная).
- *двунаправленная* – процесс может выполнять одновременно как передачу, так и получение данных (*дуплексная*).
- *полудуплексная* – в каждый момент времени может выполняться либо получение, либо передача.

Буферизация – временное хранение данных перед их окончательной передачей.

Адресуемые данные организуются по принципу *FIFO*. Для передачи сообщений может использоваться как *FIFO*, так и *LIFO*.

Алгоритмы управления процессами

First-Come, First-Served (FCFS)

Простейшим алгоритмом планирования является алгоритм, который принято обозначать аббревиатурой *FCFS* по первым буквам его английского названия – First-Come, First-Served (первым пришел, первым обслужен).

Представим себе, что процессы, находящиеся в состоянии готовность, выстроены в очередь. Когда процесс переходит в состояние готовность, он помещается в конец этой очереди. Выбор нового процесса для исполнения осуществляется из начала очереди с удалением его оттуда.

Очередь подобного типа имеет в программировании специальное наименование – *FIFO*, сокращение от First In, First Out (первым вошел, первым вышел).

Round Robin (RR)

Модификацией алгоритма FCFS является алгоритм, получивший название *Round Robin* (Round Robin – это вид детской карусели в США) или сокращенно *RR*. По сути дела, это тот же самый алгоритм, только реализованный в режиме вытесняющего планирования.

Можно представить себе все множество готовых процессов организованным циклически – процессы сидят на карусели. Карусель вращается так, что каждый процесс находится около процессора небольшой фиксированный квант времени, обычно 10 – 100 миллисекунд. Пока процесс находится рядом с процессором, он получает процессор в свое распоряжение и может исполняться.

Shortest-Job-First (SJF)

Короткие задачи в начале очереди.

Алгоритм может быть как *вытесняющим*, так и *невытесняющим*. При *невытесняющем* SJF - планировании процессор предоставляется избранному процессу на все необходимое ему время, независимо от событий, происходящих в вычислительной системе. При *вытесняющем* SJF - планировании учитывается появление новых процессов в очереди готовых к исполнению во время работы выбранного процесса.

Если CPU burst нового процесса меньше, чем остаток CPU burst у исполняющегося, то исполняющийся процесс вытесняется новым.

CPU burst – промежуток времени непрерывного использования процессора.

Гарантированное планирование

При интерактивной работе N пользователей в вычислительной системе можно применить алгоритм планирования, который гарантирует, что каждый из пользователей будет иметь в своем распоряжении $\sim 1/N$ часть процессорного времени. Пронумеруем всех пользователей от 1 до N. Для каждого пользователя с номером i введем две величины: Ti – время нахождения пользователя в системе или, другими словами, длительность сеанса его общения с машиной и – суммарное

процессорное время уже выделенное всем его процессам в течение сеанса. Справедливым для пользователя было бы получение Ti/N процессорного времени. Если

$$\tau_i << T_i/N$$

то i -й пользователь несправедливо обделен процессорным временем. Если же

$$\tau_i >> T_i/N$$

то система явно благоволит к пользователю с номером i. Вычислим для процессов каждого пользователя значение коэффициента справедливости

$$\tau_i N/T_i$$

и будем предоставлять очередной квант времени готовому процессу с наименьшей величиной этого отношения. Предложенный алгоритм называют алгоритмом *гарантированного планирования*. К недостаткам этого алгоритма можно отнести невозможность предугадать поведение пользователей. Если некоторый пользователь отправится на пару часов пообедать и поспать, не прерывая сеанса работы, то по возвращении его процессы будут получать неоправданно много процессорного времени.

Статические и динамические характеристики процессов

Статические – не изменяются в ходе выполнения процесса:

- предельное кол-во ресурсов в системе
- каким пользователем/процессом запущено
- приоритет
- сколько процессорного времени запрошено
- соотношение cpu-birst/io-birst
- какие ресурсы необходимы и в каком количестве

Динамические – не известные на этапе загрузки:

- количество свободных ресурсов
- turnaround time
- waiting time
- response time
- сколько прошло времени с момента выгрузки на диск
- сколько RAM потребляет
- cpu-birst
- io-burst

Семафоры и Mutex

Семафор – специальный механизм, указывающий процессам, средствам связи и ресурсам их состояние.

Mutex – однобитный семафор. Обычно указывает состояние готовности.

Классическая задача. Обедающие философы

Пять безмолвных философов сидят вокруг круглого стола, перед каждым философом стоит тарелка спагетти. Вилки лежат на столе между каждой парой ближайших философов.

Каждый философ может либо есть, либо размышлять. Приём пищи не ограничен количеством оставшихся спагетти — подразумевается бесконечный запас. Тем не менее, философ может есть только тогда, когда держит две вилки — взятую справа и слева (альтернативная формулировка проблемы подразумевает миски с рисом и палочки для еды вместо тарелок со спагетти и вилок).

Каждый философ может взять ближайшую вилку (если она доступна), или положить — если он уже держит её. Взятие каждой вилки и возвращение её на стол являются раздельными действиями, которые должны выполняться одно за другим.

Суть проблемы заключается в том, чтобы разработать модель поведения (параллельный алгоритм), при котором ни один из философов не будет голодать, то есть будет вечно чередовать приём пищи и размышления.

Проблемы

Задача сформулирована таким образом, чтобы иллюстрировать проблему избежания взаимной блокировки (deadlock) — состояния системы, при котором прогресс невозможен.

Например, можно посоветовать каждому философу выполнять следующий алгоритм:

- Размышлять, пока не освободится левая вилка. Когда вилка освободится — взять её.
- Размышлять, пока не освободится правая вилка. Когда вилка освободится — взять её.
- Есть
- Положить левую вилку
- Положить правую вилку
- Повторить алгоритм сначала

Это решение задачи некорректно: оно позволяет системе достичь состояния взаимной блокировки, когда каждый философ взял вилку слева и ждёт, когда вилка справа освободится.

Официант

Относительно простое решение задачи достигается путём добавления официанта возле стола. Философы должны дожидаться разрешения официанта перед тем, как взять вилку. Поскольку официант знает, сколько вилок используется в данный момент, он может принимать решения относительно распределения вилок и тем самым предотвратить взаимную блокировку философов. Если четыре вилки из пяти уже используются, то следующий философ, запросивший вилку, вынужден будет ждать разрешения официанта — которое не будет получено, пока вилка не будет освобождена. Предполагается, что философ всегда пытается сначала взять левую вилку, а потом — правую (или наоборот), что упрощает логику. Официант работает, как семафор.

Иерархия ресурсов

Другое простое решение достигается путём присвоения частичного порядка ресурсам (в данном случае вилкам) и установления соглашения, что ресурсы запрашиваются в указанном порядке, а возвращаются в обратном порядке. Кроме того, не должно быть двух ресурсов, не связанных порядком, используемых одной рабочей единицей.

Пусть ресурсы (вилки) будут пронумерованы от 1 до 5, и каждая рабочая единица (философ) всегда берёт сначала вилку с наименьшим номером, а потом вилку с наибольшим номером из двух доступных. Далее, философ кладёт сначала вилку с большим номером, потом — с меньшим. В этом случае, если четыре из пяти философов одновременно возьмут вилку с наименьшим номером, на столе останется вилка с наибольшим возможным номером. Таким образом, пятый философ не сможет взять ни одной вилки. Более того, только один философ будет иметь доступ к вилке с наибольшим номером, так что он сможет есть двумя вилками. Когда он закончит использовать вилки, он в первую очередь положит на стол вилку с большим номером, потом — с меньшим, тем самым позволив другому философу взять недостающую вилку и приступить к еде.

Классическая задача. Спящий брадобрей

Проблема

Аналогия основана на гипотетической парикмахерской с одним парикмахером. У парикмахера есть одно рабочее место и приемная со многими стульями. Когда парикмахер заканчивает подстригать клиента, он отпускает клиента и затем идет в приёмную, чтобы посмотреть, есть ли ждущие клиенты. Если есть, он приглашает одного из них и стрижет его. Если ждущих клиентов нет, он возвращается к своему креслу и спит в нем.

Каждый приходящий клиент смотрит на то, что делает парикмахер. Если парикмахер спит, то клиент будит его и садится в кресло. Если парикмахер работает, то клиент идет в приёмную. Если есть свободный стул в приёмной, клиент садится и ждёт своей очереди. Если свободного стула нет, то клиент уходит. Основываясь на наивном анализе, вышеупомянутое описание должно гарантировать, что парикмахерская функционирует правильно с парикмахером, стригущим любого пришедшего, пока есть клиенты, и затем спящим до появления следующего клиента. На практике есть много проблем, которые могут произойти, которые иллюстрируют общие проблемы планирования.

Все проблемы связаны с фактом, что все действия и парикмахера, и клиента (проверка приёмной, вход в парикмахерскую, занятие места в приёмной, и т. д.) занимают неизвестное количество времени. Например, клиент может войти и заметить, что парикмахер работает, тогда он идет в приёмную. Пока он идет, парикмахер заканчивает стрижку, которую он делает и идет, чтобы проверить приемную. Так как там никого нет (клиент еще не дошел) он возвращается к своему месту и спит. Парикмахер теперь ждет клиента, и клиент ждет парикмахера. В другом примере два клиента могут прибыть в то же самое время, когда в приемной есть единственное свободное место. Они замечают, что парикмахер работает, идут в приёмную, и оба пытаются занять единственный стул.

Решение

Доступно множество возможных решений. Основной элемент каждого — *mutex*, который гарантирует, что изменить состояние (*state*) может только один из участников. Парикмахер должен захватить это *mutex* исключение, прежде чем проверить клиентов, и освободить его, когда он начинает или спать, или работать. Клиент должен захватить *mutex*, прежде чем войти в магазин, и освободить его, как только он займет место или в приемной, или у парикмахера. Это устраняет обе проблемы, упомянутые в предыдущей секции. *Семафоры* также обязаны указывать на состояние системы. Например, можно было бы сохранить число людей в приемной.

У варианта с несколькими парикмахерами есть дополнительная сложность координирования нескольких парикмахеров среди ждущих клиентов.

Классическая задача. Читатели и писатели

Дана некоторая разделяемая область память, к этой структуре данных может обращаться произвольное количество «читателей» и произвольное количество «писателей».

Несколько читателей могут получить доступ одновременно, писатели в этот момент не допускаются. Только один писатель может получить доступ, другие писатели и читатели должны ждать.

Первое решение

Читатель может войти в критическую секцию, если нет писателей.

Это решение несправедливо, так как отдает предпочтение читателям

Плотный поток запросов от читателей может привести к тому, что писатель никогда не получит доступа к критической секции – ситуация «голодания» (starvation).

Второе решение

Отдадим предпочтение писателям, то есть читатель не входит в критическую секцию, если есть хотя бы один ожидающий писатель.

Данное решение отдает приоритет писателям, и тоже несправедливо, так как возможно «голодание» (starvation) читателей.

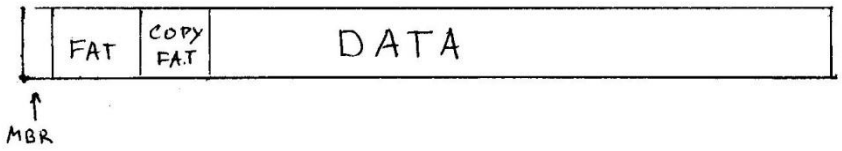
Третье решение

Не отдавать никому приоритета, просто использовать *мьютекс*.

Файловые системы

FAT - File Allocation Table

- FAT12 для дискет
- FAT16 диски до 2гб
- UFAT можно использовать нелатинские символы в названиях
- FAT32 имя хранится в заголовке файла



MBR – главная загрузочная запись (размер кластера, место загрузки ОС)

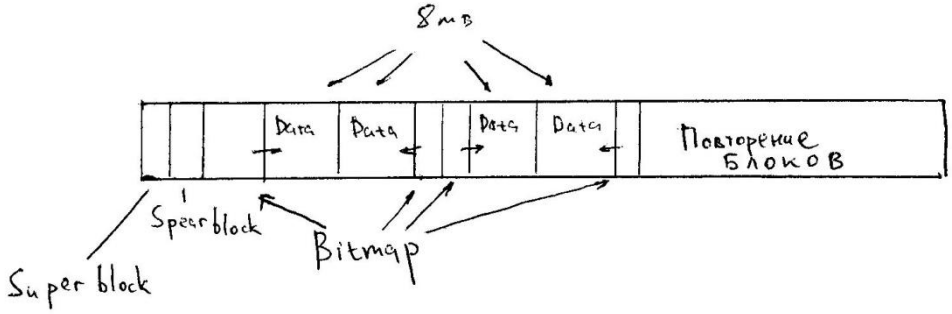
FAT – таблица разметки.

- имя файла
- свойства файлов
- размер
- начальная позиция на диске

В последнем байте при фрагментации файла – адрес след. фрагмента файла.

Каждый следующий фрагмент в начале имеет ссылку на предыдущий фрагмент.

HPFS – High Performance File System

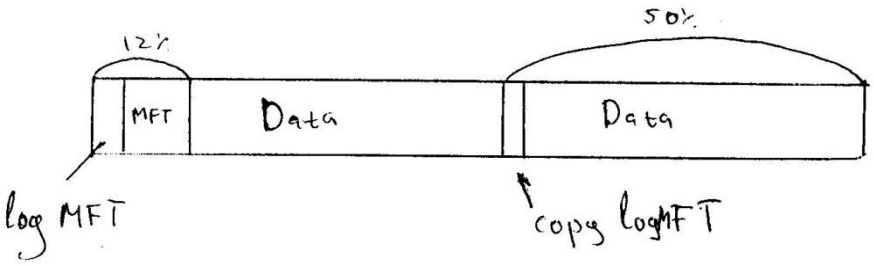


Super block – аналог главной загрузочной записи

Spear block – информация в процентном отношении о занятости каждой битовой карты (картотека битовых карт)

Bitmap – битовая карта, аналог FAT, показывает, какие данные записаны (размер 8мб)

NTFS – New technologies file system



Log MFT – загрузочная запись, аналог mbr

MFT – ссылка на адрес журнала событий (записи о каждом файле на диске(имя, владелец, создатель, атрибуты, дата последнего изменения, наличие мягких и твердых ссылок) размером 1 Кб, или любые другие файлы размером до 1кб). Занимает 12%. Если не хватает, то система будет сдвигать границу.

Элементы графического интерфейса

В большинстве существует стандартный набор элементов интерфейса, включающий следующие элементы управления:

- кнопка (button)
- радиокнопка (radio button)
- флажок (check box)
- значок (иконка, icon)
- список (list box)
- дерево — иерархический список (tree view)
- раскрывающийся список (combo box, drop-down list)
- поле редактирования (textbox, edit field)
- элемент для отображения табличных данных (grid view)
- меню (menu)
- главное меню окна (main menu)
- контекстное меню (popup menu)
- ниспадающее меню (pull down menu)
- окно (window)
- диалоговое окно (dialog box)
- модальное окно (modal window)
- панель (panel)
- вкладка (tab)
- панель инструментов (toolbar)
- полоса прокрутки (scrollbar)
- ползунок (slider)
- строка состояния (status bar)
- всплывающая подсказка (tooltip, hint)

Парадигмы текстовых процессоров. Издательские системы

4 *парадигмы*:

1. символ
2. слово
3. абзац
4. документ

В издательской деятельности пятый элемент – страница.

Параметр символа

- шрифт

Параметры шрифта:

- гарнитура (наименование шрифта в соответствии с типом оформления). Для текста есть основные гарнитуры: пропорциональный шрифт с засечками (Times), пропорциональный без засечек (Arial легко сканируется), моноширинный с засечками (Courier). Также есть специальные: символически (симбол) и пиктографические (окрыленный))
- кегль. 1 пт = (1/72)''. 1'' = 2.54см
- начертание **ж** *к* ч x^2 x_2
- кернинг – расстояние между серединами букв.

Параметр слова:

- язык

Параметры абзаца:

- отступ/выступ
- междустрочный интервал
- положение на странице (всегда целиком, запрет висячих строк, не отрывать от следующего, запрет переносов)
- табуляция – фиксированный отступ от полей листа (списки, оглавление, таблицы) = 1,27см

Параметры документа:

- параметры разделов страниц, их оформления колонтитулы
- сноски
- пояснения в конце страницы.

Что можно делать с документом:

- набор текста
- редактирование текста – изменение

- форматирование – задание оформления

Стиль – именованное задание параметров шрифта, абзаца.

Правила оформления документов

- Пробел после любого знака препинания
- В конце заголовков точки не ставятся
- Кавычки не отделяются пробелами от слов
- Перенос слов осуществляется автоматически

Shift+Enter – принудительный переход на новую строку, без образования абзаца.

Парадигмы электронных таблиц, правила построения формул, адресация

Парадигмы

- ячейка - пересечение столбца и строки
- лист - большая таблица
- книга - много листов

Формулы

Любая формула начинается с =, иначе программа будет пытаться определить тип данных.

Дальше идут имена ячеек, значения и функции, объединенные арифметическими функциями.

- (;) перечисление
- (:) диапазон

3% преобразуется 0,03

Часто используемые функции

- =Сумм(диапазон)
- =Корень(a)
- =Степень(a;b) a^b
- =Остат(делимое;делитель)
- =Целчасть(делимое;делитель)
- =Округл(y;x) до значения x(x=0 до целого)
- =Мин(диапазон)
- =Макс(диапазон)
- =Срзнач(диапазон)
- =Счет(диапазон) кол-во непустых ячеек
- =Если(условие; если 1; если 0)
- =И(a;b) логическое умножение
- =Или(a;b) логическое сложение
- =Дата(год();месяц();день())
- =Сегодня()
- =Деньнед()
- =суммесли(диапазон поиска; значение сравнения; диапазон значений; режим сравнения)

Дата является числом. Точка отсчета: *01.01.1900*

Правила создания презентаций

Этапы создания:

1. подбор материалов
2. формирование структуры
3. представление информации
4. оформление слайдов
5. настройка анимации

Правила:

- использовать короткие слова и предложения
- время глаголов должно быть согласовано
- заголовки должны привлекать внимание
- предпочтительно горизонтальное расположение информации
- наиболее значимая информация должна быть посередине в центре экрана
- подписи к картинкам должны быть под ними
- не следует заполнять 1 слайд лишней информацией
- человек может запомнить не более 3 фактов, выводов, определений одновременно
- наибольшая эффективность достигается тогда, когда ключевые пункты отражаются каждый на отдельном слайде.
- надо придерживаться единого стиля, дизайна и оформления.
- надо избегать отвлекающего от смысла презентации оформления и анимации

Шрифты:

- заголовки не менее 24pt
- остальные 18
- шрифты без засечек легче читать с большего расстояния
- не следует смешивать разные типы шрифтов в одной презентации
- не злоупотреблять заглавными буквами

Не более трех цветов на одном слайде

- фон
- заголовки
- текст

Для фона использовать спокойные оттенки.

Анимация:

В меру, не злоупотреблять, анимировать только необходимые элемент

Анимация не должна отвлекать от информации.

Компьютерная сеть, её компоненты, серверы и станции

Компьютерная сеть – несколько или более компьютеров, объединенных между собой коммуникационными системами и разделяющих общие ресурсы.

Компоненты сети

- *Серверы* –программы или устройства, которые дают в общее пользование какой-либо ресурс.
- *Станции* – обычные компьютеры в сети
- *Коммуникационные системы*

Сервера

- *Выделенный* (dedicated) – компьютерное устройство, занимающееся только предоставлением ресурсов.
- *Невыделенный* (non-dedicated) – компьютер и устройство, работающее, как сервер и как обычная машина.

Станции

- *Дисковые* – сетевое программное обеспечение и система грузятся с локального диска
- *Бездисковые* – частичная загрузка с локального диска

Сетевая коммуникационная система. Устройства и кабели

- *Концентратор* (Hub) – устройство, позволяющее обмениваться информацией между компьютерами.
- *Коммутатор* (Switch) – почти то же самое. Запоминает адрес отправителя, приходящий пакет отправляется только тому, кому он предназначен.
- *Multi Access Unit* – посылает пакеты по кругу (в два направления)
- *Repeater* – устройство, повторяющее сигнал, для его усиления
- *Мост* (Bridge) – устройство, соединяющее две подсети с одной коммуникационной системой.
- *Маршрутизатор* (Router) – устройство, соединяющее несколько подсетей с различной коммуникационной системой
- *Шлюз* (Gate Way) – тот же маршрутизатор, но работает с протоколами высокого уровня

Кабели

- Тонкий коаксиал (BNC соединение). Для построения сетей применяются BNC коннекторы, T, I коннекторы, 50-омные заглушки
- Витая пара. Для обжимания концов используется RJ45 коннектор

Одноранговые сети. Преимущества, недостатки

Одноранговая сеть – сеть, основанная на равноправии участников. В такой сети отсутствуют серверы, а каждый узел является как клиентом, так и сервером.

Преимущество одноранговых сетей заключается в том, что разделяемыми ресурсами могут являться ресурсы всех компьютеров в сети и нет необходимости копировать все используемые сразу несколькими пользователями файлы на сервер. Любой пользователь сети имеет возможность использовать все данные, хранящиеся на других компьютерах сети, и устройства, подключенные к ним.

Затраты на организацию одноранговых вычислительных сетей относительно небольшие. Однако при увеличении числа рабочих станций эффективность их использования резко уменьшается.

Основной недостаток работы одноранговой сети заключается в значительном увеличении времени решения прикладных задач. Это связано с тем, что каждый компьютер сети отрабатывает все запросы, идущие к нему со стороны других пользователей. Следовательно, в одноранговых сетях каждый компьютер работает значительно интенсивнее, чем в автономном режиме.

Существует еще несколько важных проблем, возникающих в процессе работы одноранговых сетей: возможность потери сетевых данных при перезагрузке рабочей станции и сложность организации резервного копирования.

Сети с выделенным сервером. Преимущества, недостатки

Сеть с выделенным сервером — это локальная вычислительная сеть, в которой сетевые устройства централизованы и управляются одним или несколькими серверами. Индивидуальные рабочие станции или клиенты должны обращаться к ресурсам сети через сервер(ы).

Преимущества

Сервер спроектирован так, чтобы предоставлять доступ к множеству файлов и принтеров, обеспечивая при этом высокую производительность и защиту.

Администрирование

Администрирование и управление доступом к данным осуществляется централизованно. Ресурсы, как правило, расположены также централизованно, что облегчает их поиск и поддержку.

Резервное копирование данных

Поскольку жизненно важная информация расположена централизованно, т.е. сосредоточена на одном или нескольких серверах, нетрудно обеспечить ее регулярное резервное копирование (backup).

Избыточность

Благодаря избыточным системам данные на любом сервере могут дублироваться в реальном времени, поэтому в случае повреждения основной области хранения данных информация не будет потеряна — легко воспользоваться резервной копией.

Количество пользователей

Сети на основе сервера способны поддерживать тысячи пользователей. Сетями такого размера, будь они одноранговыми, было бы невозможно управлять.

Аппаратное обеспечение

Так как компьютер пользователя не выполняет функций сервера, требования к его характеристикам зависят от потребностей самого пользователя. Типичный компьютер-клиент имеет, по крайней мере, 486-й процессор и от 8 до 16 Мб оперативной памяти.

Недостатки

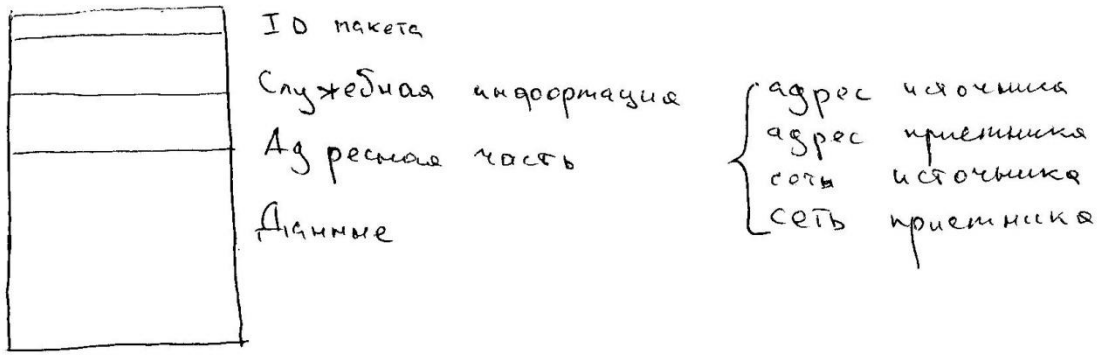
Сеть не будет работать, при выходе сервера из строя.

Необходимость покупки сервера.

Протоколы передачи данных. Устройство сетевого пакета

Протокол – условный язык передачи данных по сети и организации адресации.

Устройство пакета



Примеры протоколов

- TCP/IP (IPv4, IPv6). Адреса имеют вид x.x.x.x (x∈ $[0..255]$), для IPv4. Для IPv6 адрес имеет вид x:x:x:x:x:x:x:x (x∈ $[0..FFFF]$)
- IPX/SPX. В качестве адреса IPX использует идентификатор, образованный из четырёхбайтного номера сети (назначаемого маршрутизаторами) и MAC-адреса сетевого адаптера.
- NetBEUI. Протокол локальных сетей от Microsoft. Адреса устанавливаются администраторами устройств. Пример: HOMESTATION

Критерии классификации сетей. Топология

Классификация сетей

- По скорости передачи (<10 мб – низк., 10-100 мб – средн., 10 мб – 1 гб – выс., >1 гб сверх. скоростные)
- По типу используемого кабеля
- По типу соединения кабеля
- По типу передаваемых пакетов

Топология – один из видов классификации системы.

Топологии соединения сетевых устройств

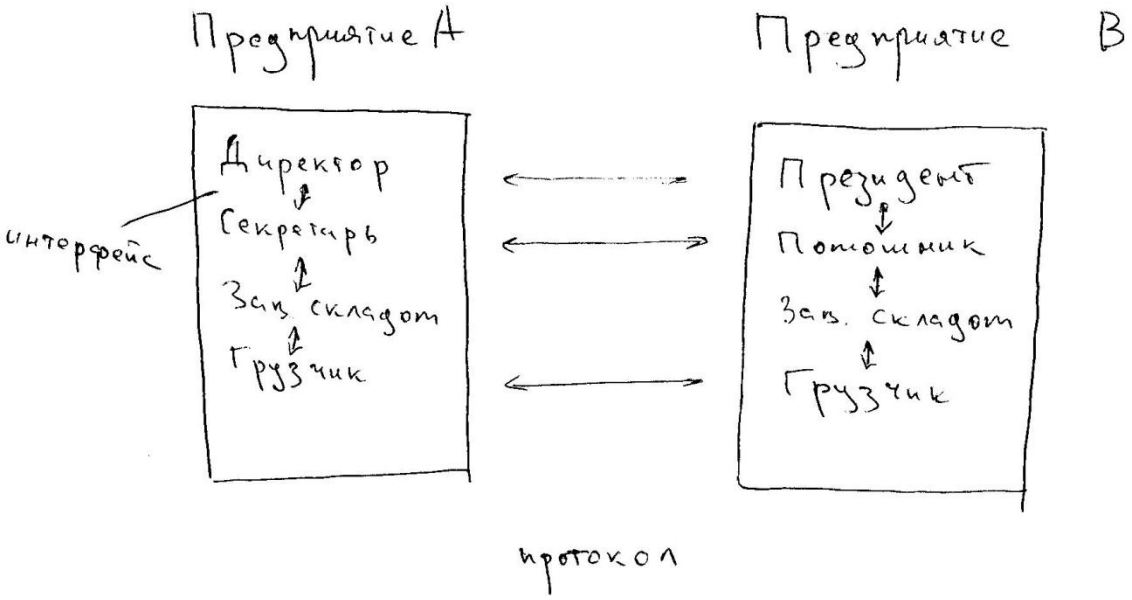
- *Bus* (шина) – общий кабель, к которому подключены все компьютеры. Медленная, самая простая. Максимальная длина кабеля между крайними станциями 185 метров. Максимальная длина кабеля между двумя соседними машинами – 60 метров
- *Star* (звезда) – все компьютеры сети подключены к концентратору или коммутатору. Относительно быстрая, надежная
- *Ring* (кольцо) – каждый компьютер соединен только с двумя другими. Сигнал идет в определенном направлении. Простота установки. Можно использовать маркер (один маркер на сеть, вещать может тот, у кого есть маркер, маркер передается)
- *Mesh* (ячеистая) – каждый компьютер соединен с несколькими другими. Сложная, надежная, быстрая, дорогая
- *Hybrid* (смешанная) – совмещение предыдущих

Модели OSI

OSI-open system interconnection.

В 1982 году Международная организация по стандартизации (ISO) в сотрудничестве с ITU-T начала новый проект в области сетевых технологий, названный взаимодействием открытых систем, Open Systems Interconnection или OSI.

До OSI сетевые технологии были полностью проприетарными. OSI стала новой попыткой создания сетевых стандартов для обеспечения совместимости решений разных поставщиков. В то время многие большие сети были вынуждены поддерживать несколько протоколов взаимодействия и включали большое количество устройств, не имеющих возможность общаться с другими устройствами из-за отсутствия общих протоколов.



7.прикладной уровень(Application layer) – данные, информация

6.представительный уровень (Presentation layer) – данные, форматы файлов

5.сеансовый уровень (Session layer) – данные на уровне разделения памяти

4.транспортный уровень (Transport layer) – блоки данных(контроль надежности и достоверности)

3.сетевой уровень (Network) – пакеты данных(образуются сетевые адреса)

2.канальный уровень (Data link) – кадры(разбиение на пакеты и их ID)

1.физический уровень (Physical) – биты(электрические, оптические, электромагнит. сигналы)