МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени Н.Э. БАУМАНА

Учебное пособие

А.Е.Аверьянихин

Курс лекций

«Системное программирование»

МГТУ имени Н.Э. Баумана

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
имени Н.Э. БАУМАНА

А.Е.Аверьянихин

Курс лекций

«Системное программирование»

Москва
МГТУ имени Н.Э. Баумана

2012

УДК 681.3.06(075.8)
ББК 32.973-018
И201

А.Е.Аверьянихин
Курс лекций «Системы искусственного интеллекта» / под ред. И.М.Холина –
М.: МГТУ им. Н.Э. Баумана, 2012. – 26 с.: ил.

В курсе лекций рассмотрены основные этапы курса «Системное программирование».

Ил. 39. Табл. 5. Библиогр. 7 назв.

УДК 681.3.06(075.8)

АННОТАЦИЯ

В курсе лекций будут рассмотрены основные темы курса «Системное программирования» такие как: представление данных в вычислительных системах, основы работы с ОС Linux теории алгоритмов и программирования на языке С.

СОДЕРЖАНИЕ

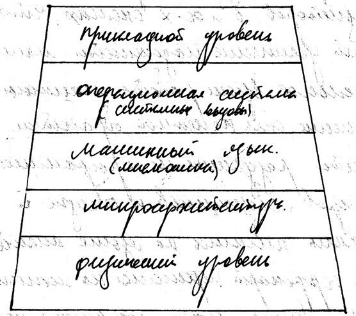
ВВЕДЕНИЕ

Данный конспект лекций составлен на основе лекционного курса, читаемого в МГТУ им. Н.Э. Баумана на кафедре ИУ4 преподавателем Аверьянихиным А.Е. Курс лекций рекомендован к выполнению текущих аттестационных мероприятий и подготовки к зачету по предмету «Системное программирование».

Лекции

Лекция №1 2012-09-05

Вычислительная сис-ма — это совокупность аппаратно-программных узлов, взаимодействующих между собой. Их многообразие обуславливает широкое разнообразие методов управления. Стремление к унификации описания машин решение мат-х задач привело к появлению понимания вычислительной системы с точки зрения многоуровневой модели:



ОС можно рассматривать с 2-х точек зрения:

1) расширенная виртуальная машина, методы управления набором стандартизированы.
POSIX — описывает семейство ОС на базе Unix.
2) ОС как менеджер ресурсов. С этой точки зрения ОС предоставляет пользователю методы управления информацией.

В развитии ОС выделяют 4 поколения.

1) 1945 - 1955 г.г. - поколение характеризуется применением электронных ламп. Машины однозадачные, однопользовательские. Рабочий цикл состоит из решения 1 задачи. Описание логики выполняется с помощью переключателей или электронных панелей. Пионер: Фон Нейман. Завершением этапа принято считать появление перфокарт.

2) 1955 - 1965 - этап характеризуется применением транзисторов в лог-х схемах, что обеспечивает увеличение надежности машин. Длительное время работы машины обусловило применение схем пакетной обработки. Появляется разделение на программистов, операторов, сборщиков и обслуж-й персонал. Программы писались на языке ассемблер. Набор перфокарт переносили на магнитную ленту. Лента могла содержать несколько заданий, которые выполнялись в едином цикле работы машин.

3) 1965 - 1980 - характеристики поколения: ИС, многозадачность. Впервые предприняты попытки создания схожих по архитектуре и обслуживанию вычислительных систем под надписью IBM/360

Для управления этой линейкой машин впервые была начата разработка системы OS/360, которая могла управлять всей линейкой. Впервые была применена попытка многозадачности на 1-процессорной системе. Она заключалась в том что, память разделялась на набор блоков фиксированного размера. Блок занимала 1 задача. Выполнение задачи происходило до операции, требующей периферийную операцию. Задачи выполнялись циклически. Периферийные операции — spooling.

Начало развития многозадачности привело к созданию разделения во времени. Этот мех-зм основан на циклической обработке выполнения задач. Каждой задаче выделялся определенный квант процессорного времени, по истечении которого задача приостанавливалась. Реализовано в CTSS (совместимая система разделения времени). Была попытка создания большой ЭВМ на MULTICs. Предполагалось увеличение систем больших ЭВМ и появление популярных [illegible] мини-машин без УДХ (фирма DEC PDP-семейство).

4) 1980 - наши дни — характеризуется созданием процессора и линейки ПК IBM PC под MS-DOS, которая ставилась на любую совместимую машину. Середина 1980-х — создание граф.-го интерфейса для PC.

ОС Unix

Разрабатывалась как ОС для университетов. Исходные коды распространялись бесплатно до появления ПК. AT&T после появления ПК начала распространять Unix на платной лицензии. Это создало предпосылку создания в ряде университетских центров клонов Unix, таких как BSD и System V. В 1991 на базе ОС MINIX собирается 1-й Linux.

Лекция №2 | 2012-09-19

Представление данных в вычислительных системах. Машинные языки, адресация.

Алгоритмы сжатия:
1) Диффри - Хеллмана;
2) Шенона - Фано.

Архиваторы:
1) gzip;
2) tar -cvf имя_файла.tar вход - запись
   -xvf - извлечение

Режимы адресации: Мнемоника [адрес] [данные]

Множество вычислительных схем процессора обычно соответствуют некоторому набору операций, выраженному набором мнемонических команд (машинный язык). Команда может иметь поле адреса и поле данных. Существует несколько способов размещения и адресации данных, используемых в команде:

1) операнд непосредственно – данные размещаются в теле команды
   - ⊕ максимально увеличивает быстродействие;
   - ⊖ при увеличении разрядности данных увеличивается код.

   простейший вид адресации

2) операнд-регистр – данные необходимые для операции, хранятся в регистре памяти:
   - ⊕ высокое быстродействие; небольшой размер кода
   - ⊖ ограничение по объему;

3) абсолютная адресация – данные располагаются в памяти, адрес – в адресе памяти команды
   - ⊕ простой режим адресации
   - ⊖ с увеличением адресного пр-ва растет код

4) косвенно-регистровый режим – адресное поле не используется. Содержимое определенных регистров интерпретируемых как данные в памяти
   - ⊕ уменьшает код
   - ⊖ ограниченный объем

5) Косвенно-регистровая адресация – используется в сложных режимах адресации + смещение.

Адрес данных формируется содержимым регистров процесса и адресного поля команды.

⊕ Гибкость работы с данными любой разрядности; адресное поле команды используется для смещения.

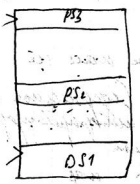
Загрузка программ:

Однозадачная система:

ОС обычно располагается в старших разрядах памяти. Для размещения используется вся память. После процесс он передает управление ОС.

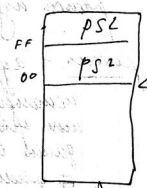
⊕ простота.

Однозадачность



1)

Многозадачность



2)

1) Абсолютная загрузка:
Процесс может загружаться с заранее определенных адресов. Свойства:
а) ограничение на макс значение загр-х процессов;
б) ограничение на объем кода.
в) явление фрагментации.

2) Относительная загрузка:
Программы загружаются с адресов, вычисляемых исходя из кол-ва уже загруженных процессов и их размеров. Накладывает ограничение на применение 5-ого режима адресации в виде невозможности определения наперед адреса данных. Для устранения этого недостатка в исполняемый код приложения вместо адресов проставляются макроопределения, которые сводятся в таблицу (relocation table – таблица перемещений), которая заполняется в момент загрузки программы, тем самым определяя конечные адреса.

Основные функции ОС. Однозадачные диспетчеры

1) Обеспечение пользовательского интерфейса;
2) —||— интерфейс с устройствами;
3) загрузка программ

Многозадачная ОС:

4) управление памятью;
5) псевдопараллельное выполнение задач;
6) организация межзадачных взаимодействий;
7) защита процессов друг от друга (сложный режим адресации)
8) мультипользовательность

1) DOS - Disk Operation System (1 задача, 1 пользователь)
2) Общего назначения
3) Операционные системы реального времени - сбрасо регламентированные сроки загрузки.
4) Операционные системы виртуальных машин

Лекция №3 | 2011-10-03

Управление памятью.

В сложном режиме адресации (№5) адрес состоит из 2-х частей:

1) база
2) смещение

база смещение

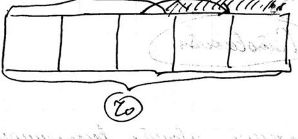
№.0 линейное пространство опер-й памяти делится на:

1) страницы;
2) сегменты.

Адрес начала блока - адрес базы. Смещение - значение, которое прибав. к адресу базы. УПП - обеспечивает страницную адресацию памяти для процессов, расширяя доступную область адресации за счет устройств

одновременного хранения. Механизм размещения информации в ЗУ — механизм страничного планирования.

Кольца защиты процессора — аппаратно-реализованные ограничения на адресное пространство памяти, используемые для защиты процессов от влияния друг на друга. Кольца защиты (режим работы процессора) переключаются через прерывания. Современная ОС использует 2 кольца защиты.



Внутр. фрагментация — внутри страниц и сегментов

Внешняя — // — всего адресного пространства памяти

Процессы в кольце r_0 — процессы, выполняемые в режиме ядра (супервизора) (kernel mode)
Процессы в кольце r_1 — процесс в режиме пользователя (user mode)

Процессы

Процесс — абстракция выполняемой программы. Процесс загружается в свое адресное пространство.

2 способа создания процессов:

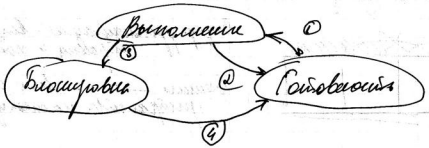
1) инициализация системы;
2) клонирование:

род-й процесс вызывает for, система дублирует процесс, exec вызывает замену сегмента кода и сброс сегмента данных из стека.

Завершение процессов:

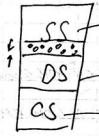
1) нормальное завершение процессов (добровольное);
2) завершение вследствие ошибки (добровольное);
3) завершение вследствие фатальной ошибки (принудительное);
4) уничтожение процесса другим процессом (принудительное) — только родитель может убить потомка, но не наоборот.

Модель состояний процессов.



Процесс выполняется в течение кванта выделенного времени. При реализации многозадачности процесс, приостановленный таск менеджер переводит в состояние готовности (1), а любой в центре (1). Процесс может выполнять блокировку (ожидание завершения периферийной операции) (3). По завершению ожидания операции процесс возвращается в режим готовности (4). Далее таск менеджер переводит его на выполнение.
Для поддержания упорядоченной структуры процессов и хранения доп. инфо-ии используется таблица процессов (таблица 5)

Состав процесса



SS — stack segment — выделяется ОС, но они не пересекаются
DS — data segment — инициируется процессом
CS — code segment

bash

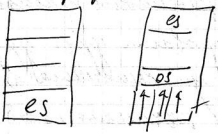
while (command) – //blog команд
if (fork (command) != 0) // клонирование процесса
exec (command) // замена процесса

fork PID ?
↓
потомку родителю

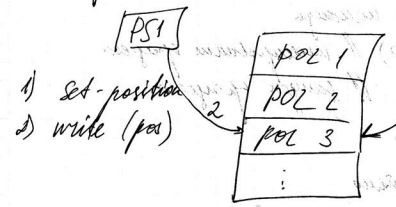
Лекция №4 | 2012-10-17

Взаимодействие между процессами

Программные каналы



В многозадачных сис-ах внутри 1 процесса может быть несколько потоков управления. Это необходимо для параллельной обработки данных внутри 1 процесса (например: web-браузер). Управление потоками может быть реализовано на уровне ОС или библиотеки. Как правило библиотечное управление быстрее. Однако не позволяет управлять составными потоков на уровне ОС (пример: процессу посылается сигнал блокировки).



Реализация многозадачности на однопроцессорных сис-мах имеет 3 аспекта:

1) передача инф-ии от одного процесса другому (реализуется на уровне ОС через сервер передачи сообщений);
2) проблема взаимоисключения. Взаимодействующие процессы не должны нарушать целостность данных, с которыми работают и блокировать друг друга.
3) согласование действий процессов при выполнении задач (принтер печатает не должен начинать печать, до того как файл, выводящий данные не закончит писать в spooling).

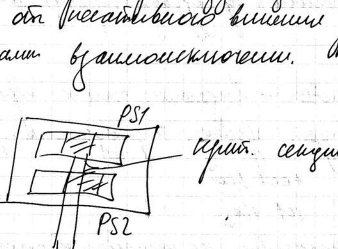
Проблема взаимного влияния на разделяемые данные.
2 процесса одновременно пишут в стек данные.
Запись состоит из этапов:

1) получение позиции в стеке;
2) запись данных.

Допустим пр. 1 выполняет получение позиции в стеке. После этого планировщик приостанавливает его работу; пр. 2 выполняет операцию получения поз-ии в стеке и получает то же самое, производит запись данных, после чего возвращается процесс 1. Далее пр. 1 пишет свои данные поверх данных пр. 2. Целостность данных нарушена.

<u>Критическая секция</u> – участок кода в котором процесс работает с разделяемыми данными.

Алгоритм, который реализует защиту данных с процессов от негативного влияния друг друга наз-ся алгоритмами взаимоисключения.



Перетерабельный ресурс – такой блок, который гарантирует получение верного результата независимо от скорости и очередности выполнения.

Неперетерабельный блок ничего не гарантирует.

Примитивы взаимоисключения

1) Самый грубый способ – запретить прерывание процесса в момент его нахождения внутри критической секции. Данная реализация имеет след. недостаток: процесс может получить доступ к ресурсам, которые длительно время находятся его приоритет;

2) алгоритм строгого чередования. Заключается в перемещении использования разделяемых ресурсов строго поочередно. Недостаток: невозможность предоставить требуемую систему обращения;

3) Перемещая блокировки – выделение спец. переменных, которые контролируют доступ к разделяемым ресурсам.

Работа с флагами разделяется на несколько этапов.

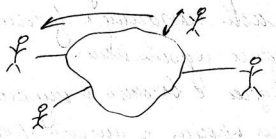
Предположив что процесс одинаково быстро работает с флагом и всегда запрет на прерывание процесса во время работы с флагом (неделимая операция) получаем работоспособный механизм взаимоисключения, который имеет след-ие недостатки:

1) допустим 2 процесса с высоким и низким приоритетами. Допустим, процесс с низким приоритетом получил управление, установил флаг, выполнил операцию с ресурсами и был остановлен планировщиком. Управление получает процесс с высоким приоритетом и начинает проверять флаг. Выполняет эту операцию в течение всего времени. Т.о. процесс с высоким приоритетом нах-ся в ожидании процесса с низким приоритетом. Это называется инверсией приоритетов. В следующий момент процесс с высоким приоритетом получит меньше времени;

2) активное ожидание: пр. с ↑ приоритетом выполняет проверку флага. Операция проверки флага нагружает сис-му. Процесс получивший объект и заблокированный к флагу блокируется сам. Генерирует след. паузу во времени и повторяет запрос по его истечении. Т.о. процесс с низким приоритетом получает возможность завершить все процессы не напрягая сис-му. Пример: CDMA (collision detect multiple access).

Лекция №5 | 2012-10-31

Вероятность



Процесс обмена информацией между 2 участниками в открытой сети с точки зрения информационной безопасности существует ряд факторов, которые могут оказывать влияние на защищенный процесс обмена пакетами:

1) информация может быть получена 3-им лицом (man in middle);

2) информация может быть модифицирована 3-им участником открытой сети;

3) информация может быть не доступна 1 из уч-в.

Уязвимость - потенциальная возможность злоумышленника нарушить работу обмена данных.

Угроза - реализация уязвимости. Правильно в неформальной информационной безопасности

Атака - реализация угрозы, активная деятельность злоумышленника, направленная на нарушение защищенного обмена данными.

Виды атак:

1) DDOS;
2) man in a middle;
3) направленная отсыл ложных DNS ответов;
4) IP spoofing (подмена IP-адреса)
5) Brood force (грубая сила)

Аспекты защиты информации

1) Оптимальная конфигурация сетевой инфраструктуры;
2) Применение криптографических алгоритмов для шифрования информации.

Виды криптографических алгоритмов

1) Потоковый (шифр Цезаря) - самая простая реализация;
2) Блочный (Субстанс) более требовательный к аппаратному обеспечению;
3) Функции сверки-однонаправленные функции.

Любой может защитить от угрозы раскрытия и модификации путем шифрования. Однако возникает необходимость организации защиты канала передачи данных.

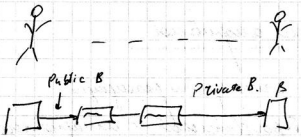
При разработке алгоритма безопасного обмена ключами принята к классу алгоритмов, которые используют 2 ключа — для шифрования и расшифровки.

Алгоритмы шифрования

- Симметричные: DES, 3
- Ассиметричные: RSA, DSA

Ассиметричный алгоритм обмена информацией

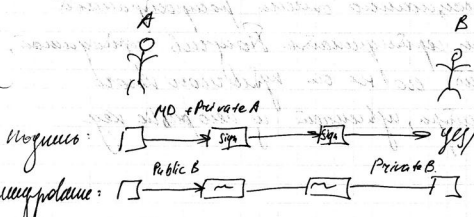
Public B	Public A
Private A	Private B
Public A	Public B



Ассиметричный шифр. взаимодействует генерируя взаимодополняющую пару ключей: для шифрования, расшифровки. Шифруется Public'ом, расшифровывается Privat'ом, не наоборот.

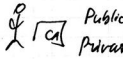
Описанный метод имеет уязвимости при подтверждении личности абонентов.

Для подтверждения личности используют механизм цифровой подписи, основанный на односторонних функциях.



Описанный метод двухключевой с т. зрения компрометации участников взаимодействия не требует создания ключей.

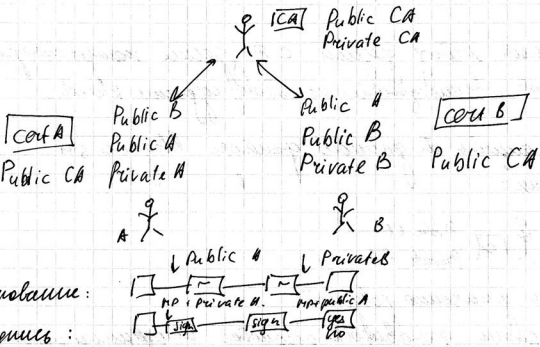
Для защиты от данного фактора вводится 3-й участник:



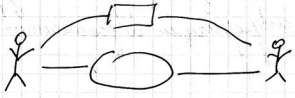
Ему по умолчанию доверяют.
Public ключи доставляются абсолютно надежными путями.
Public ключи для участников взаимодействия не распространяются в открытом виде.
Вместо него каждый участник подписывает свой ключ, т.е. получает свой сертификат.

Сертификат – публичный ключ участника, подписанный закрытым ключом удостоверяющего центра, и содержит данные определенной информации об участниках обмена информацией.

Участники информационного обмена распространяют друг между которым сертификаты. Получив сертификат, участник проверяет его на ось публичного ключа удостоверяющего центра, извлекает из него public key другого участника.



Описанная схема является нашей схемой информационного обмена, однако ассиметричные алгоритмы не рек. шифр. больших объемов данных. Поэтому они используются для передачи ключей симметричных алгоритмов шифрования.



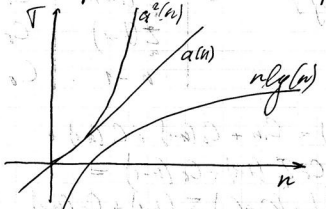
Алгоритм. Анализ

Алгоритм - строго определенная последовательность простых действий, обеспечивающая решение задачи за конечное число.

С т. зрения оптимизации алгоритмов введены 2 вида сложности:

- вычислительная (зависимость времени выполнения об входа);
- пространственная (зависимость памяти об входа).

Вычислительная сложность - это степень полинома, которым выражается время выполнения об размерности входа.



С т. зрения решения одной и той же задачи прибегать к разным алгоритмам. Например, сложность сортировки вставками - n^2, а быстрой сортировки $n \lg(n)$, т.е. она эффективнее на больших массивах. При анализе сложности алгоритмов следует учитывать наихудший случай входа. Например, алгоритм сортировки

пузырьком имеет сложность и при работе с уже отсортированным массивом n^2 при работе с любыми другими массивами.

Расчёт сложности алгоритма основывается на анализе псевдокода алгоритма. Для составления псевдокода необходимо каждую абстрактную операцию расписать на более сложные, опустив тривиальные вычисления.

Анализ алгоритма сортировки вставками

	T — кол-во опер.	C — время опер.
for $j \leftarrow 2$ do length [A]	n	C_1
do key $\leftarrow$ A[j]	$n-1$	C_2
$i = j - 1$	$n-1$	C_3
while $i>0$ and A[i] > key	$\sum_2^n t$	C_4
do A[i+1] = A[i]	$\sum_2^n (t-1)$	C_5
$i = i - 1$.	$\sum_2^n (t-1)$	C_6
A[i+1] = key	$n-1$	C_8

Сложность алгоритма: $T(n) = C_1 n + C_2(n-1) + C_4(n-1) + C_3 \sum_2^n t + C_4 \sum_2^n (t-1) + C_5 \sum_2^n (t-1) + C_6(n-1) =$

$= C_1 n + 2C_2(n-1) + C_3 \sum_2^n t + (C_4 + C_5) \sum_2^n (t-1) + C_8(n-1).$

$$\sum_2^n t = \frac{n(n+1)}{2} - 1$$

$$\sum_2^n (t-1) = \frac{n(n-1)}{2}$$

В наилучшем случае алгоритм будет выполняться n раз. Cn ; C - совокупность операций. В общем случае while будет выполняться некоторое время. Формула ряд. Анализ сортировки методом вставки в общем случае имеет сложность n^2.

Анализ сложности сортировки методом "пузырька."

	C	T
for j=1 To N-1	C_1	n
FOR i=1 to N-j	C_1	n^2
IF A[i]>A[i+1]	C_2	n^2-1
THEN swap A[i], A[i+1]	C_3	$\frac{1}{2}$
NEXT i	C_4	n^2
NEXT i	C_4	n

Всегда расбет как n^2

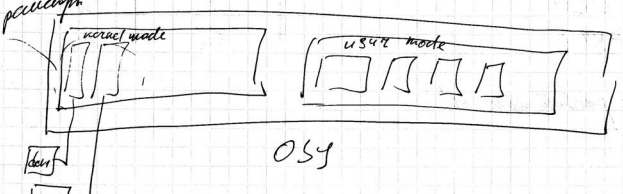
Ввод-вывод

С точки зрения ОС устр-во представляет собой набор регистров и буферов.

Буферы - используются для обмена данными;

Регистры - для управления работой уст-в и диагностики его состояния.

ОС проверяет состояние регистров во время взаимодействия с устр-м.



Обычно регистры и буферы устр-в отображены на область памяти, которая в целях защиты доступна в режиме ядра. Взаимодействие с устр-м реализовано в обычном режиме с обычными адресами. Взаимодействие может выполняться ЦП, однако в силу скоростных ограничений устройств ресурс процессорного времени не является оптимальным.

Для того, чтобы процессор мог решать полезные задачи в современных системах сущ-т контроллер DMA (Direct Memory Access). Процессор осуществляет его программирование, отвлекается от системной шины, позволяя выполнять операции по обмену данными, вычислению checksum, ошибок.

ПО ввода-вывода состоит из след-х уровней:

1) прерывание — набор программ по обработке событий генерируемых устройствам;
2) драйверы — управляющие устройством программы, которые:
   a) взаимодействует с картой памяти устр-ва;
   b) регистрирует и исправляет ошибки взаим-я данных;
   c) регулирование скоростью обмена данными.
   d) предоставление единообразных интерфейсов вызова для программ верхнего уровня (драйвер сис-мы)
3) аппаратнонезависимый код ОС — предоставляет программам пользовательского уровня единообразной интерфейс доступа к базовым функциям. Код ОС (Unix) на этом уровне представляет устр-во как файл, который может быть: файл символьного устр-ва (по 8 бит), файл блочного устр-ва (большими блоками). Как правило символьные устр-во сам-но генерирует данные (не устройства хранения данных). На этом уровне каждый файл устройства имеет только 1 или св-во · 2 номера: старший и младший. Старший — признак, который связывает файл с конкретным драйвером. Младший номер —

используется для определения начала секции драйвера для того или иного уст-ва - например. в одном устройстве 2 устройства имеющих разные начала, они выполняются одним драйвером. Будут 2 файла с 1 старшим и разными младшими.

4) пользовательское ПО (как правило библиотеки), например <stdio.h> языка C.