Системы программирования.

лекции 25.04.2011

Стратегии распределения памяти.
Типы трансляторов.

Системы программирования: компоненты и схема работы
Интегрированная среда разработки программных продуктов.

Задача распределения памяти

конструкция ЯП

– код пользовательской прогр.
– данные программы
– коды системных программ, обслуж. выполнение польз. прогр.
– записи о текущем состоянии процесса (об активации процедур...)

каждой конструкции ЯП в соответствие должны быть поставлены

– адрес памяти
– размер
– атрибуты

Память (по способу использования): локальная и глобальная.
Память (по способу распределения): статическая и динамическая.

Статическое распределение памяти

Самая простая стратегия.

Размеры объектов, размещенных статически, не меняются.

Относительные адреса (внутри области) не изменяются.

Перед загрузкой программы в память устанавливается начальный адрес статической области.

Эта стратегия используется для кодов пользоват. программ, буферов ввода / вывода
глобальных и статических переменных,
внутренних структур данных (таблица виртуальных функций)

Динамическое распределение памяти

Выбирается для случаев, когда на этапе компиляции не удается определить положение объекта в некоторой области памяти и/или его размер.

Способы распределения:

стек,

куча (произвольное распределение)

Используется при распределении памяти

под локальные переменные,

для хранения динамических объектов,

для записи о текущем состоянии процесса выполнения прогр.

Динамическое распределение памяти (прод.)

Явное выделение памяти

– блоками фиксированного размера.
Блоки памяти связываются в список, над которым достаточно легко выполнять операции их выделения и освобождения.
Достоинства: если программа полностью использует блок памяти, то нет никаких дополнительных расходов.
С каждым блоком связан указатель на его начало, требуется хранить только информацию о том, занят блок или нет.

– блоками переменного размера.
Размер блока выбирается на основе параметров запроса пользователя, хранится в самом блоке.
Возможна проблема из-за фрагментации памяти.

Динамическое распределение памяти (прод.)

Неявное выделение / освобождение памяти

выполняется компилятором и программами системной поддержки (при отсутствии явно записанных операций управления памятью в программе пользователя)

Блоки обычно имеют следующую структуру:

- **размер блока (если блок фиксированного размера — не хранится);**
- **пометка занятости либо счетчик ссылок:**
- **указатели на блоки**
- **память, выделяемая по запросу пользователя**

Счетчик ссылок — количество указателей в программе, которые ссылаются на этот блок (при присваивании указателей, например, *p* = *q*, счётчик для блока, на который указывал указатель *p* уменьшается на единицу, а для *q* увеличивается). Если счётчик становится равным нулю, то блок освобождается.

Существует проблема циклических ссылок, когда счётчики всегда >0.

Пометка фиксирует, есть хотя бы один указатель, ссылающийся на этот блок.

В некоторый момент начинает работать «сборщик мусора».

Он помечает все блоки как недостижимые, а потом начинает анализ текущих указателей программы, и находит действительно используемые блоки, достижимые по указателям.

Типы трансляторов

Компилятор

на вход получает программу на некотором ЯП (немашинном), а на выходе выдает объектный модуль (программу на машинном языке).

Pascal, gcc

Интерпретатор

на вход получает программу на некотором ЯП (немашинном) и, считывая предложение за предложением исходной программы, анализирует их и тут же выполняет действия, указанные в этих предложениях.

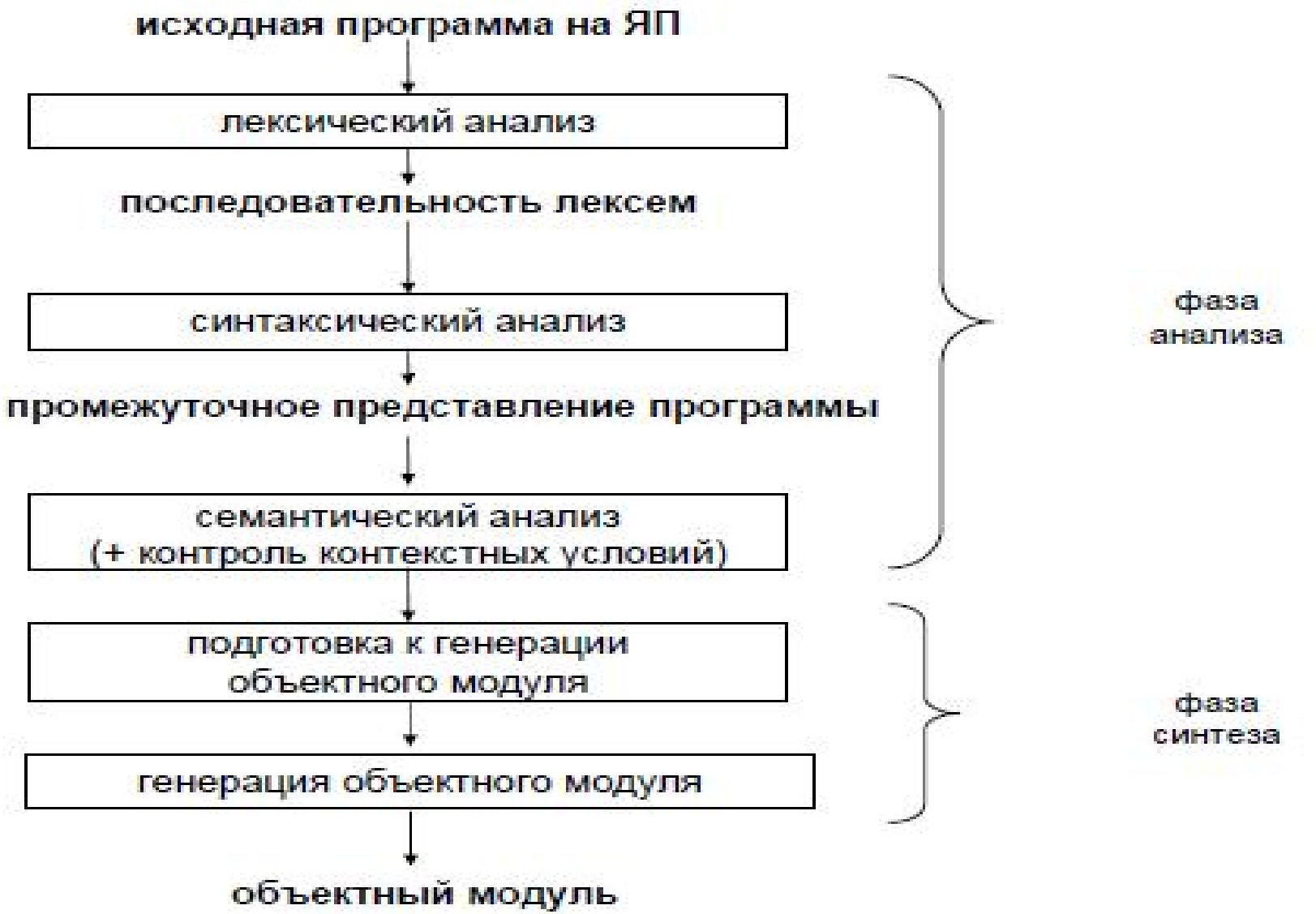
bash, csh

Смешанная статегия трансляции

Результатом работы компилятора является программа на промежуточном языке, ее и исх. данные получает на вход интерпретатор.

Java

Логическая схема работы компилятора



Система программирования

Система программирования — это комплекс программных средств (инструментов, библиотек), предназначенных для поддержки разработки программного продукта на протяжении всего жизненного цикла этого продукта.

В чем разница между программой и программным продуктом?

Система программирования

Система программирования — это комплекс программных средств (инструментов, библиотек), предназначенных для поддержки разработки программного продукта на протяжении всего жизненного цикла этого продукта.

В чем разница между программой и программным продуктом?

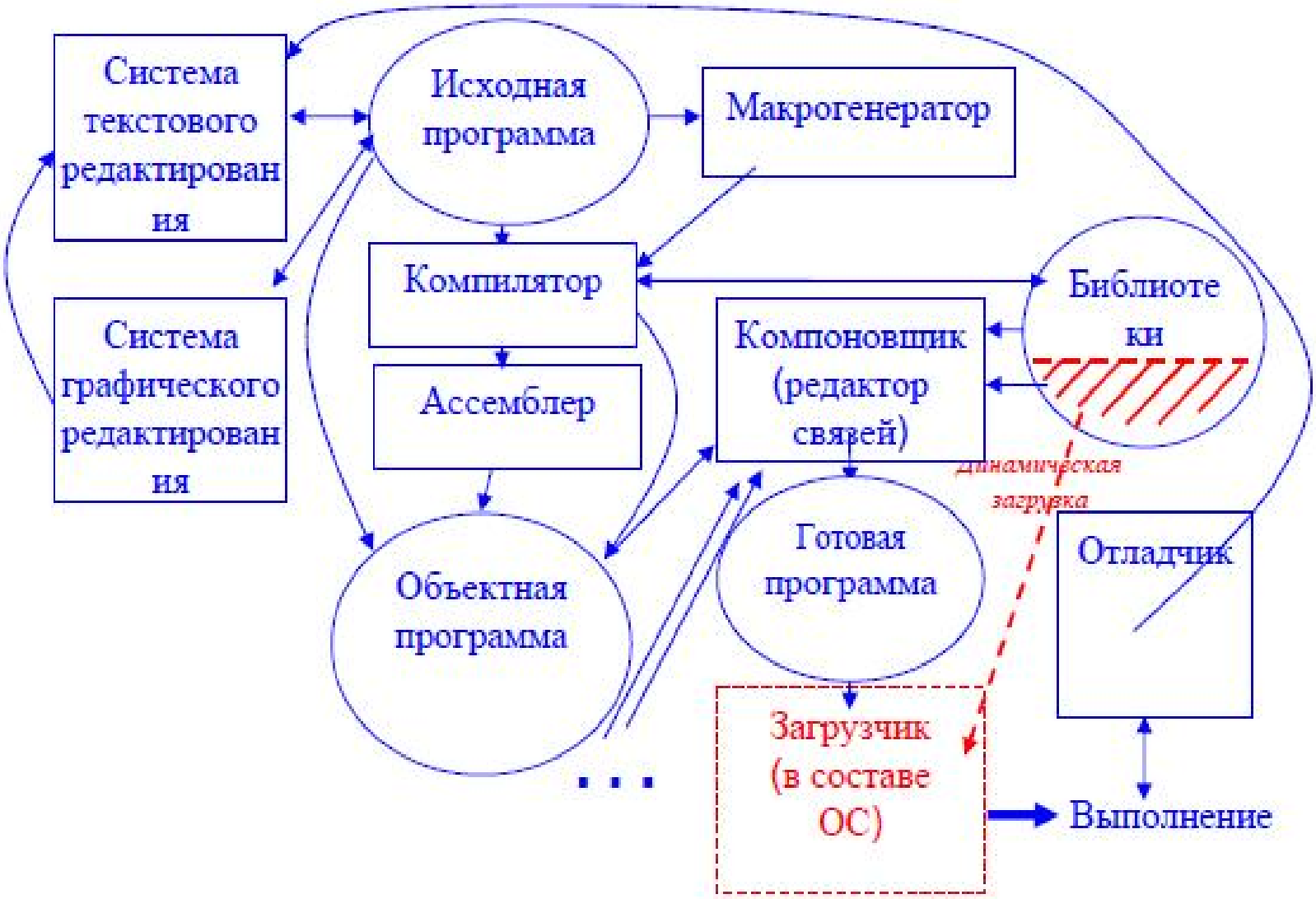
Программа создается автором для решения конктетной задачи в конкретной ОС

Программный продукт — программа, которая работает без авторского надзора в рамках некоторого набора ОС, обладает «дружественным» интерфейсом, имеет пользовательскую и техническую документацию.

Этапы жизненного цикла программного продукта

1) Постановка задачи. Анализ (определение) требований
2) Проектирование
3) Написание текста программ (программирование, “кодирование”)
4) Компоновка или интеграция программного комплекса
5) Верификация, тестирование и отладка
6) Документирование
7) Внедрение
8) Тиражирование
9) Сопровождение, повторяющее все предыдущие этапы

Состав и схема функционирования классической СП



Дополнительные требования, поддерживаемые СП

согласованность (изменения в одном месте должны инициировать изменения там, где используется измененный фрагмент ПП),

непротиворечивость (принимаемые решения на одном из этапов должны быть согласованы с решениями, принимаемыми на других этапах создания ПП),

корректность изменений (вносимые изменения не должны нарушать общие требования к ПП),

хранение "истории" развития проекта (предыдущие версии),

возможность работы группы разработчиков в реальном времени,

автоматическое или автоматизированное кодирование и документирование ПП,

автоматизацию календарного планирования сроков создания ПП.

Интегрированная среда разработки

Интегрированная среда разрабоки (ИСР) - комплекс программных средств, поддерживающих полный жизненный цикл ПП, объединяющий возможности компонент СП.

Простая ИСР объединяет в себе возможности текстовых редакторов исходных текстов программ, отладчиков и командный язык компиляции

Усовершенствованная ИСР содержит
– *репозиторий*.
Он должен обеспечивать хранение версий проекта и его отдельных компонентов, синхронизацию поступления информации от различных разработчиков при групповой разработке, контроль данных о проекте на полноту и непротиворечивость;
– *графические средства анализа и проектирования*,
обеспечивающие создание и редактирование иерархически связанных диаграмм, образующих модели ПП

Редактор текстов в рамках ИСР

Решаемая задача — подготовка текстов программ.

Интегрированность с компилятором

Интегрированность с отладчиком
– отображение контрольных точек останова при отладке,
– отображение текущего значения объекта, при наведении курсора на идентификатор.

Отладчики в рамках ИСР

Отладчик — программа, помогающая анализировать поведение отлаживаемой программы, обеспечивая ее трассировку.

Основные функции отладчика в рамках ИСР:

1) пошаговое выполнение программы **(**шаг **=** строка**;** с трассировкой внутри вызываемой функции и без нее**),**

2) выполнение программы до строки**,** в которой в редакторе стоит курсор**,**

3) выделение выполняемой в данный момент строки**,**

4) приостановка выполнения программы:

-- можно запросить значение переменной**,**

-- можно выполнить вычисление некоторого выражения**,**

-- можно изменить значение переменной и продолжить выполнение программы**,**

5) расстановка**/** снятие точек останова**,** которые показываются в текстовом редакторе**,**

6) выдача всей информации в терминах исходной программы**.**

Редактор внешних связей

Решаемая задача: связывание между собой (по внешним данным) объектных файлов, порождаемых компилятором, а также файлов (статических) библиотек, входящих в состав СП.

Создает таблицы:
1) для трансляции относительных адресов (она используется загрузчиком) ;
2) для точек вызова функций динамически подключаемых библиотек.

Загрузчик

(в современных вычислительных системах обычно является частью ОС)

Решаемая задача: преобразование относительных адресов в абсолютные адреса непосредственно в момент запуска программы на выполнение (трансляция адресов).

Профилировщики

Профилировщик строит «профиль» программы, где выделены линейные участки кода (фрагменты программы, где нет передачи управления), с указанием времени и частоты их исполнения.

Используется для оптимизации программы :
редко исполняемые линейные участки можно не оптимизировать, сосредоточив основные усилия на оптимизации критичных циклических участков.

Позволяет получать информацию о вызовах функций ядра ОС, прерываниях, состояниях потоков ввода/вывода, сообщениях и деятельности планировщика.

В состав операционных систем UNIX входит профилировщик prof.

Библиотеки. Основные типы библиотек

Библиотеки функций

Определяют возможности системы. Разрабатываются для ЯП и для решения задач в конкретной предметной области. Представляют собой откомпилированные объектные модули.

Библиотеки классов

Все классы пишутся на том же языке, что и программа. Классы — конкретные, абстрактные, шаблоны– включаются в программу на этапе компиляции.

Библиотеки компонент

Откомпилированные программные модули, которые можно использовать в качестве составной части программы, и которыми можно манипулировать во время разработки программы. Компоненты бывают локальные (находящиеся на том же компьютере, где создается прикладная программа) и распределенные (расположенные на сервере и доступные по сети). Примеры: генераторы отчетов, построители сводных таблиц, компоненты для построения графиков.

Критерии проектирования стандартных библиотек

Требования по свойствам компонентов:

– общезначимость содержимого

– эффективность

– безопасность

– завершенность

– сочетаемость с базовыми типами данных

– могут служить основой создания других библиотек.

Способы подключения библиотек функций

Статические библиотеки

Подключаются к программам, готовящимся к выполнению, ровно один раз в момент формирования редактором связей полной программы.

Динамические библиотеки

– Компоненты динамических библиотек подключаются к программам во время выполнения этих программ. Компоненты динамических библиотек не связаны с программам и, которые к ним обращаются, и распространяются отдельно от них. На этапе компоновки программы редактор связей формирует таблицу точек вызова функций библиотеки для последующей операции динамического связывания (то есть компоновщик не помещает в программу тела функций).

Системы программирования в вычислительной системе