Санкт-Петербургский государственный университет

телекоммуникаций имени профессора М.А. Бонч-Бруевича

Курсовая работа

по дисциплине Информатика

на тему:

**Трансляция, линкование, компиляция, интерпретация**

Выполнила: студентка 1 курса

Царыгина К.О.

Преподаватель: Левчук Ю.П.

2009 год

**Содержание**

Введение

Трансляция, компиляция, интерпретация, линкование

1. Трансляция. Компиляция. Интерпретация

2. Линкование

Заключение

Используемая литература

**Введение**

Большая часть работы программистов связана с написанием исходного кода, тестированием и отладкой программ на одном из языков программирования. Различные языки программирования поддерживают различные стили программирования. Единственный язык, напрямую выполняемый процессором — это *машинный язык* (также называемый *машинным кодом*). Изначально все программисты прорабатывали программы в машинном коде, но сейчас эта трудная работа уже не делается. Вместо этого программисты пишут исходный код на языке программирования высокого уровня, и компьютер (используя компилятор, интерпретатор или ассемблер) транслирует его, в один или несколько этапов, уточняя все детали, в машинный код, готовый к исполнению на целевом процессоре. Запись исходных текстов программ при помощи языков программирования облегчает понимание и редактирование человеком. Этому, в частности, помогают комментарии, допустимые в синтаксисе большинства языков. Для выполнения на компьютере готовый текст программы преобразуется (компилируется) в машинный код. Некоторые языки программирования позволяют обходиться без предварительной компиляции программы и переводят её в инструкции машинного кода непосредственно во время исполнения. Этот процесс называется динамической компиляцией, и он позволяет добиться большей переносимости программ между разными аппаратными и программными платформами при сохранении многих плюсов компиляции.

Интерпретируемые программы, для которых, как правило, не применяется процесс компиляции и которые интерпретируются операционный системой или специальными программами-интерпретаторами, называются скриптами или «сценариями».

Технология программирования задач различается для операторных и функциональных языков программирования. Ограничимся рассмотрением вопроса для операторных языков. Тогда этот этап разбивается на два последовательных шага – разработку алгоритма и отладку программы.

Отладка программы – это самый трудоемкий этап. Его цель – проверка синтаксической и логической правильности программы, а также определение того, что программа функционирует на всем диапазоне допустимых данных.

В процессе отладки программы выделяются этапы:

1. трансляция исходного текста программы;
2. компоновка программы;
3. выполнение программы с целью определения логических ошибок;
4. тестирование программы

**Трансляция, компиляция, интерпретация, линкование**

**1. Трансляция. Компиляция. Интерпретация**

**Трансляция** программы — преобразование программы, представленной на одном из языков программирования, в программу на другом языке и, в определённом смысле, равносильную первой. При трансляции выполняется перевод программы, понятной человеку, на язык, понятный компьютеру. Выполняется специальными программными средствами (транслятором).

**Трансляторы** реализуются в виде компиляторов или интерпретаторов. С точки зрения выполнения работы компилятор и интерпретатор существенно различаются. Если цель трансляции – преобразование всего исходного текста на внутренний язык компьютера (т.е. получение некоторого нового кода) и только, то такая трансляция называется также **компиляцией**. Исходный текст называется также исходной программой или исходным модулем, а результат компиляции – объектным кодом или объектным модулем. Если же трансляции подвергаются отдельные операторы исходных текстов и при этом полученные коды сразу выполняются, такая трансляция называется **интерпретацией**. Поскольку трансляция выполняется специальными программными средствами (трансляторами), последние носят название компилятора или интерпретатора, соответственно.

Цель трансляции — преобразовать текст с одного языка на другой, который понятен адресату текста. В случае программ-трансляторов, адресатом является техническое устройство (процессор) или программа-интерпретатор.

Виды трансляторов

Трансляторы подразделяют на:

* *Адресный*. Функциональное устройство, преобразующее виртуальный адрес в реальный адрес
* *Диалоговый*. Обеспечивает использование языка программирования в режиме разделения времени.
* *Многопроходной*. Формирует объектный модуль за несколько просмотров исходной программы.
* *Обратный*. То же, что детранслятор (декомпилятор, дизассемблер).
* *Однопроходной*. Формирует объектный модуль за один последовательный просмотр исходной программы.
* *Оптимизирующий*. Выполняет оптимизацию кода в создаваемом объектном модуле.
* *Синтаксически-ориентированный (синтаксически-управляемый)*. Получает на вход описание синтаксиса и семантики языка и текст на описанном языке, который и транслируется в соответствии с заданным описанием.
* *Тестовый*. Набор макрокоманд языка ассемблера, позволяющих задавать различные отладочные процедуры в программах, составленных на языке ассемблера

**Компиляция** — преобразование программой-компилятором исходного текста программы, написанного на языке высокого уровня в машинный язык, в язык, близкий к машинному, или в объектный модуль. Результатом компиляции является объектный файл с необходимыми внешними ссылками для компоновщика.

Компилятор читает всю программу целиком, делает ее перевод и создает законченный вариант программы на машинном языке, который затем и выполняется.

Виды компиляции

* *Пакетная*. Компиляция нескольких исходных модулей в одном пункте задания.
* *Построчная*. То же, что и интерпретация.
* *Условная***.** Компиляция, при которой транслируемый текст зависит от условий, заданных в исходной программе. Так, в зависимости от значения некоторой константы, можно включать или выключать трансляцию части текста программы.



Рисунок 1. Компиляция размеченного документа

**Интерпретация** — процесс непосредственного покомандного выполнения программы без предварительной компиляции, «на лету»; в большинстве случаев интерпретация намного медленнее работы уже скомпилированной программы, но не требует затрат на компиляцию, что в случае небольших программ может повышать общую производительность.

Типы интерпретаторов

*Простой интерпретатор* анализирует и тут же выполняет (собственно интерпретация) программу покомандно (или построчно), по мере поступления её исходного кода на вход интерпретатора. Его достоинство - мгновенная реакция. Недостаток — такой интерпретатор обнаруживает ошибки в тексте программы только при попытке выполнения команды (или строки) с ошибкой.

*Интерпретатор компилирующего типа* — это система из компилятора, переводящего исходный код программы в промежуточное представление, например, в байт-код или p-код, и собственно интерпретатора, который выполняет полученный промежуточный код (так называемая виртуальная машина). Его достоинство – большее быстродействие выполнения программ (за счёт выноса анализа исходного кода в отдельный, разовый проход, и минимизации этого анализа в интерпретаторе). Недостатки — большее требование к ресурсам и требование на корректность исходного кода.

Алгоритм работы простого интерпретатора

1. прочитать инструкцию;
2. проанализировать инструкцию и определить соответствующие действия;
3. выполнить соответствующие действия;
4. если не достигнуто условие завершения программы, прочитать следующую инструкцию и перейти к пункту 2.

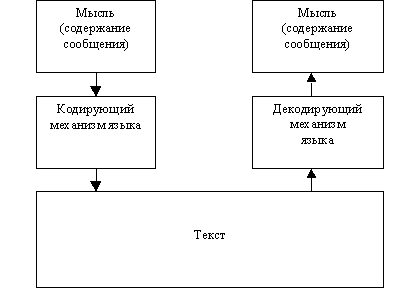


Рисунок 2. Схема интерпретации текста Ю. Лотмана

**2. Линкование**

**Линкование (компоновка)** - это процесс, при котором все "недокомпилированные" части программы доводятся до конца и связываются между собой в исполняемый файл (или файлы) формата, понятного данной операционной системе. В итоге, мы получаем исполняемую программу.

Линкование важно, так как при разработке больших проектов код обычно вырастает настолько, что приходится его для повышения надежности раскидывать по разным файлам.

Для связывания модулей компоновщик использует таблицы имён, созданные компилятором в каждом из объектных модулей. Такие имена могут быть двух типов:

* *Определённые* или *экспортируемые* имена — функции и переменные, определённые в данном модуле и предоставляемые для использования другим модулям
* *Неопределённые* или *импортируемые* имена — функции и переменные, на которые ссылается модуль, но не определяет их внутри себя

Работа компоновщика заключается в том, чтобы в каждом модуле разрешить ссылки на неопределённые имена. Для каждого импортируемого имени находится его определение в других модулях, упоминание имени заменяется на его адрес.

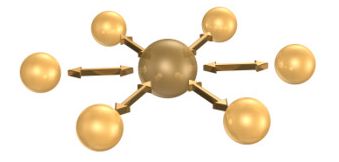


Рисунок 3. Наглядная схема линкования

Существует два способа создания линкования:

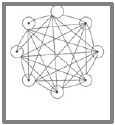
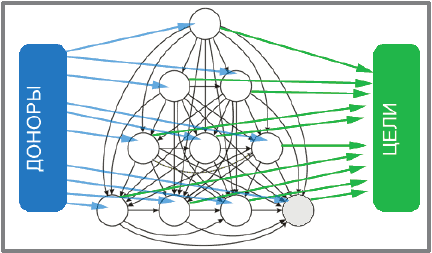


Рисунок 4. Схема «Кольцо»

Рисунок 5. Схема «Елочка»

**Заключение**

Упрощая, можно сказать, что при компилировании перевод программы в понятную для машины форму осуществляется сразу же после её создания. То есть из исходного текста на языке высокого уровня получается машинный код, пригодный для исполнения на данном процессоре. Если необходимо выполнить программу на другом типе процессора с несовместимой системой команд, то необходима перекомпиляция исходной программы на языке высокого уровня для данного типа процессора.

Программа же на интерпретируемом языке транслируется в машинные коды лишь при её выполнении. Делается это, по мере поступления команд, следующим образом. Каждой команде интерпретируемого языка в коде интерпретатора соответствует подпрограмма, созданная с использованием поддерживаемых аппаратной частью компьютера средств. Когда эта команда встречается в коде, связанная с ней подпрограмма запускается, и необходимые действия выполняются. Так как особенности реализации интерпретируемых языков скрыты, кажется, что имеющиеся в них команды проделываются напрямую компьютером. Поэтому соответствующие интерпретаторы называют виртуальными компьютерами (или, чаще, виртуальными машинами). Для переноса программы на другую аппаратную базу (другой процессор) необходимо лишь наличие интерпретатора данного языка программирования для данного типа процессора.

**Используемая литература**

* *Роберт У. Себеста.* Основные концепции языков программирования
* *Вольфенгаген В. Э.* Конструкции языков программирования.