

Редакторы связей. Загрузчики

Редакторы связей

- Конечным результатом компиляции является объектный модуль. За один запуск компилятора всегда порождается ровно один объектный модуль.
- Если какой-либо компилятор создает при компиляции с языка программирования текст на языке ассемблера, объектный модуль все равно создается, но работа по его созданию перекладывается на ассемблер.
- Дальнейшая работа над объектными модулями проводится **редактором связей**, которые иногда называются **компоновщиками (linker)**.

Редакторы связей

- **Основное назначение редактора связей** – завершить ту часть работы, которая принципиально не могла быть выполнена компилятором, а именно, осуществить привязку нескольких модулей друг к другу.
- *Объектный файл (или набор объектных файлов) не может быть выполнен до тех пор, пока все модули и секции не будут в нем увязаны между собой – это и делает редактор связей. Результатом его работы является единый файл («исполняемый файл»), который содержит весь текст результирующей программы на языке машинных кодов.*
- Компилятор не мог выполнить такую привязку, потому что он всегда работает только с одной компилируемой программой, зная о других программных компонентах только то, что они должны существовать, а также их программные интерфейсы

Редакторы связей

- В настоящее время программы пользователя зачастую содержат лишь незначительную часть команд и данных, необходимых для решения поставленной задачи.
- В составе системного программного обеспечения поставляются большие библиотеки подпрограмм, так что программист, которому необходимо выполнять определенные стандартные операции, может воспользоваться для этого готовыми подпрограммами.
- В частности, операции ввода-вывода обычно выполняются подпрограммами, находящимися вне программы пользователя.

Редакторы связей

- Поэтому программу на машинном языке, полученную в результате трансляции, приходится, как правило, комбинировать с другими программами на машинном языке, чтобы сформировать необходимый выполняемый модуль.
- В отличие от компилятора, который во время своего запуска обрабатывает только один объект (программный компонент), редактор связей в общем случае получает на вход сразу несколько объектных модулей.
- Эти модули могут быть получены редактором связей непосредственно от компилятора, а могут извлекаться им из библиотек, которые также указываются среди параметров запуска компоновщика.

Редакторы связей

- **Задача компоновщика проста** – он должен пройти весь код результирующей программы, начиная от места вызова ее главной исполняемой функции (точки входа) и до конца (точки выхода), найти все вызовы внешних процедур и функций, обращения к внешним переменным и увязать их с кодом других модулей, где описаны эти процедуры, функции и переменные.
- Причем в этих модулях, в свою очередь, могут быть обращения к другим внешним функциям – и т.д.
- Функции, описанные в разных модулях исходной программы, могут вызывать друг друга сколько угодно раз – компоновщик должен найти соответствие каждому из этих вызовов и **определить («разрешить») соответствующую ссылку (адрес)**. Отсюда и происходит другое название компоновщика – «редактор связей».

Редакторы связей

- Ссылки (вызовы процедур или функций, обращения к переменным), которым компоновщик не смог найти соответствие, называются «неразрешенными».
- Если же некоторые внешние связи остались неразрешенными, то есть соответствующие им объекты не обнаружались ни в одном из объектных модулей, поданных на редактирование, редактор связей выдает сообщение об ошибке.
- Редактор связей в состоянии провести и другой контроль – контроль соответствия между объектным модулем, в котором упоминается некоторое внешнее имя (используется объект с этим именем), и объектным модулем, в котором данный объект определен.

Редакторы связей

- **Задача редактора связей** – сформировать области (разделы, секции) памяти, которые впоследствии смогут быть размещены в памяти вычислительной машины как единое целое, в виде цельных блоков.
- Области могут иметь иницилирующие значения, а могут быть пустыми, то есть использоваться только для резервирования памяти.
- **Значениями, которыми иницируются разделы памяти,** могут быть последовательности команд (программные разделы) или начальные значения статических объектов данных, в том числе констант.
- **Для каждой области редактор связей вводит свой начальный адрес** (реальное его значение определяется на более поздних стадиях, вплоть до загрузки программы в оперативную память для выполнения).

Редакторы связей

- Компоновщик начинает свою работу с того, что выбирает из первого объектного модуля программную секцию и присваивает ей начальный адрес.
- Программные секции остальных объектных модулей получают адреса относительно начального адреса в порядке следования. При этом может выполняться также функция выравнивания начальных адресов программных секций.
- Одновременно с объединением текстов программных секций объединяются секции данных, таблицы идентификаторов и внешних имен.
- **Разрешаются межсекционные ссылки.**

Редакторы связей

- **Процедура разрешения ссылок** сводится к вычислению значений адресных констант процедур, функций и переменных с учетом перемещений секций относительно начала собираемого программного модуля.
- Если при этом обнаруживаются ссылки на внешние переменные, отсутствующие в списке объектных модулей, редактор связей организует их поиск в библиотеках, доступных в системе программирования.
- Если же и в библиотеке необходимую составляющую найти не удастся, формируется сообщение об ошибке.

Редакторы связей

- В современных системах программирования компоновщик включает в состав исполняемого файла не только код объектных модулей, который подготовил компилятор исходной программы, но и описание ресурсов пользовательского интерфейса, которое готовит **компилятор ресурсов**.
- Как и компилятор при распределении памяти, компоновщик работает с относительными адресами переменных, процедур и функций – условными единицами, отсчитываемыми от начала программы.
- Так же как и компилятор, он не может знать адресов памяти, в которых будет выполняться результирующая программа.

Редакторы связей

- Компоновщик работает с теми именами процедур и функций, которые им присвоил компилятор в таблице идентификаторов.
- Эти имена могут отличаться от имен, данных пользователем.
- Поэтому, если два или более обращения к одной и той же функции или переменной в исходном тексте программы выполнены по-разному, компилятор может дать им разные имена и ссылка останется неразрешенной, что приведет к сообщению об ошибке от компоновщика.
- **Другая возможная ошибка на этапе компоновки** – несоответствие между объектным файлом и файлом описания процедуры или функции.

Редакторы связей

- **Дело в том, что компилятор порождает вызовы процедур и функций, исходя из их описания.**
- Если функция описана в одном из исходных файлов, то он может проверить соответствие ее описания и исходного кода, но если обращение выполняется к библиотечной функции, то такая возможность отсутствует (библиотеки функций поставляются, как правило, в виде объектного кода).
- В том случае, когда имена функций совпадают, но описание не соответствует реальному объектному коду, может возникнуть трудно обнаруживаемая ошибка – такую ошибку можно отследить только при отладке результирующей программы. Конечно же, такие ошибки крайне редко встречаются в библиотеках систем программирования и других библиотеках процедур и функций, предоставляемых

Редакторы связей

- **От редактора связей не зависит эффективность выполнения готовой программы.**
- **Все, что можно сделать для повышения этой эффективности, делается в период компиляции.**
- Однако от редактора связей может зависеть эффективность использования памяти вычислительной машины, поскольку именно при редактировании связей определяется истинный размер готовой программы.
- Самый простой редактор связей при обнаружении ссылки на некоторый объектный модуль (или даже при передаче некоторого объектного модуля на редактирование, независимо от того, имеются на его объекты ссылки в других объектных модулях или нет) просто вставляет в готовую программу все определенные в нем объекты, как программы, так и данные.

Редакторы связей

- В таком случае файл готовой программы (и размер, занимаемой ею памяти машины) будет максимальным, и в этом файле могут оказаться объекты, никогда в программе не используемые.
- В особенности часто подобная ситуация может возникать при работе с библиотеками, в которых komponуются множество семантически связанных процедур, из которых в реальной работе используется лишь некоторая часть.
- **Чтобы избежать потерь памяти разработчики библиотек часто оформляют их не в виде одного большого библиотечного файла, а в виде набора относительно небольших файлов, вставка которых в готовые программы не будет приводить к их существующему росту**

Редакторы связей

- Однако лучшим выходом из этой ситуации является усложнение алгоритма работы редактора связей, который обладает всей информацией, необходимой для отбора тех объектов модуля, которые реально используются в программе.
 - Неиспользуемые объекты могут исключаться из рассмотрения и в формируемые разделы памяти не попадать.
 - В таком случае они не будут попадать и в файл готовой программы.
 - **Современные системы программирования стараются комплектовать именно такими редакторами связей, использование которых снижает нагрузку на**
- используемую память машины.**




Редакторы связей

- Таким образом, основные задачи редактора связей таковы:
- связывание между собой по внешним данным объектных модулей, порождаемых компилятором и составляющих единую программу;
- подготовка таблицы трансляции относительных адресов для загрузчика;
- статическое подключение библиотек с целью получения единого исполняемого модуля;
- подготовка таблицы точек вызова функций динамических библиотек

Редакторы связей

- Каждое приложение содержит список импорта, в котором перечислены используемые приложением DLL и функции, в них находящиеся.
- Во время запуска программы операционная система считывает из файла этот список из таблицы импорта и использует его для поиска адресов этих функций в указанных DLL.

Редакторы связей

IMPORTS				
  				
RVA	Name	RVA	Hint	Name
0103B298h	msvcrt.dll	01001064h	005Ah	GetLastError
0103B2A4h	ADVAPI32.dll	01001068h	0047h	CreateEventW
0103B2B2h	KERNEL32.dll	0100106Ch	005Eh	GetLocaleInfoW
0103B2C0h	GDI32.dll	01001070h	00E5h	FreeLibrary
0103B2CAh	USER32.dll	01001074h	00A3h	GetSystemDefaultLCID
0103B2D6h	ntdll.dll	01001078h	000Ah	SetProcessShutdownParameters
0103B2E0h	SHLWAPI.dll	0100107Ch	009Dh	ReleaseMutex
0103B2ECh	SHELL32.dll	01001080h	0058h	CreateMutexW
0103B2F8h	ole32.dll	01001084h	0007h	SetPriorityClass
0103B302h	OLEAUT32.dll	01001088h	002Fh	GetCurrentProcess
0103B310h	BROWSEUI.dll	0100108Ch	009Dh	GetStartupInfoW
0103B31Eh	SHDOCWV.dll	01001090h	00FEh	GetCommandLineW
0103B32Ah	UxTheme.dll	01001094h	00EBh	SetErrorMode
		01001098h	002Dh	LeaveCriticalSection
		0100109Ch	008Bh	EnterCriticalSection
		010010A0h	0031h	CompareFileTime
		010010A4h	00ACh	GetSystemTimeAsFileTime
		010010A8h	000Ch	GetSystemTime

Загрузчики

- Объектные модули строятся на основе так называемых относительных адресов.
- Компилятор, порождающий объектные файлы, а затем и компоновщик, объединяющий их в единое целое, не могут знать точно, в какой физической области памяти компьютера будет располагаться программа в момент ее выполнения.
- Поэтому они работают не с реальными адресами ячеек ОЗУ, а с некоторыми относительными адресами.
- **Такие адреса отсчитываются от условной точки, принятой за начало области памяти, занимаемой результирующей программой (обычно это точка начала первого модуля программы).**

Загрузчики

- Конечно, ни одна программа не может быть исполнена в этих относительных адресах.
- Поэтому требуется модуль, который выполнял бы преобразование относительных адресов в реальные (абсолютные) адреса непосредственно в момент запуска программы на выполнение.
- **Этот процесс называется трансляцией адресов и выполняет его специальный модуль, называемый загрузчиком.**
- **Загрузчик** — это системная программа, которая размещает команды и данные программы в ячейках основной памяти.

Загрузчики

- Однако загрузчик не всегда является составной частью системы программирования, поскольку выполняемые им функции зависят непосредственно от архитектуры целевой вычислительной системы, в которой выполняется результирующая программа, созданная системой программирования.
- *На первых этапах развития ОС загрузчики существовали в виде отдельных модулей, которые выполняли трансляцию адресов и готовили программу к выполнению – создавали так называемый «образ задачи».*

Загрузчики

- С развитием вычислительных систем появилась возможность выполнять трансляцию адресов непосредственно в момент запуска программы на выполнение.
- Для этого в состав исполняемого файла включается таблица, содержащая перечень ссылок на адреса, которые необходимо подвергнуть трансляции.
- В момент запуска исполняемого файла ОС обрабатывает эту таблицу и преобразовывает относительные адреса в абсолютные.

Загрузчики

- В современных ОС существуют сложные методы преобразования адресов, которые работают непосредственно уже во время выполнения программы.
- Эти методы основаны на возможностях, аппаратно заложенных в архитектуру вычислительных комплексов.
- Для выполнения трансляции адресов в момент запуска программы должны быть подготовлены специальные системные таблицы, с помощью которых можно определить все места в программе, где надо произвести модификацию условных или относительных адресов в абсолютные.
- **Эти функции целиком ложатся на модули ОС, поэтому они не выполняются в системах**

Загрузчики

- **Загрузчик, который выполняет трансляцию адресов в момент запуска программы, называется настраивающим загрузчиком.**
- В современных вычислительных системах такой загрузчик входит в состав ОС, а компоновщик, входящий в состав системы программирования, готовит для настраивающего загрузчика таблицу трансляции адресов, входящую в состав исполняемого файла.
- Каждая ОС содержит в себе свой настраивающий загрузчик и предусматривает свой формат таблицы трансляции адресов – система программирования при создании исполняемого файла должна учитывать это.
- **Это одна из причин, почему исполняемые файлы для различных ОС несовместимы между собой, даже если они предназначены для одной и той же архитектуры**

Загрузчики

- Вынесение загрузчиков в операционные системы имеет еще один очень важный смысл. Современные вычислительные комплексы могут иметь сложные аппаратно управляемые структуры памяти.
- **Методы трансляции адресов могут основываться на сегментной, страничной или сегментно-страничной организации памяти.**
- Полное владение ситуацией может обеспечить только операционная система, причем только в момент непосредственного занесения программы в память.
- Это означает, что загрузчик системы программирования в принципе не способен решить все проблемы модификации адресов, поскольку он не может знать точных характеристик конфигурации аппаратных средств и состояния внутренних таблиц подсистемы управления памятью операционной системы в момент, когда

Загрузчики

- При программировании на машинном языке привязка к памяти производится в момент кодирования.
- Уже давно наблюдается тенденция откладывать привязку программы к памяти на как можно более поздний срок, и в современных системах виртуальной памяти привязка осуществляется динамически в процессе выполнения программы.
- **Абсолютный загрузчик** размещает элементы программы именно в те ячейки, адреса которых указаны в программе на машинном языке.
- **Перемещающий загрузчик** может загружать программу в различные места основной памяти в зависимости, например, от наличия свободного участка основной памяти в момент загрузки.

Загрузчики

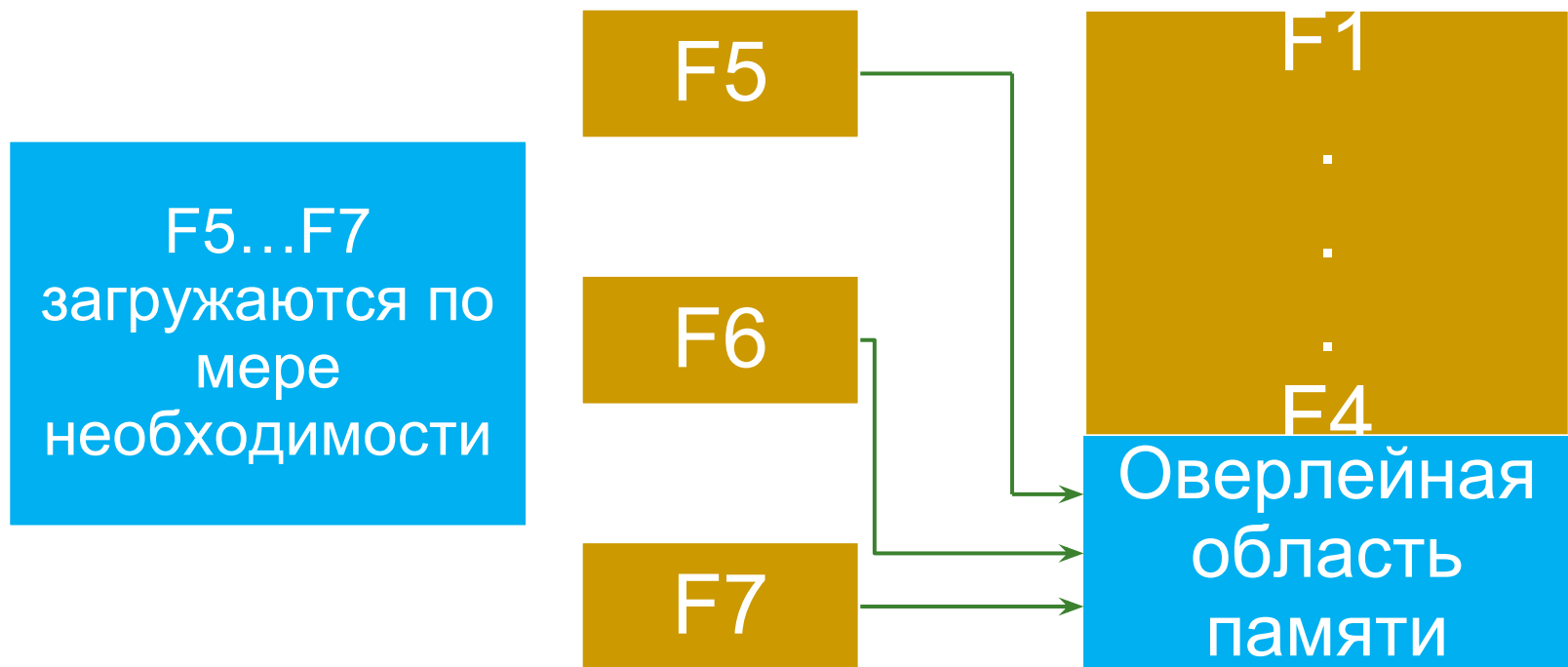
- Первоначально ОС выполняла загрузку в оперативную память компьютера полностью всей исполняемой программы вместе с кодом всех используемых в ней библиотек и модулей в момент запуска этой программы на выполнение.
- При этом, если объем программы превышал объем памяти, доступной для ее выполнения, программа не могла быть выполнена.
- Также, если код какой-нибудь библиотеки использовался в нескольких программах, всякий раз он заново загружался в оперативную память.
- Это вело к неэффективному использованию памяти.

Загрузчики

- С развитием **оверлейных структур** и **динамических библиотек** появилась возможность загружать исполняемый код модуля программы или библиотеки в оперативную память не сразу же в момент запуска программы на выполнение, а в тот момент, когда произойдет обращение к функции или процедуре, содержащейся в этом коде.
- **Оверлей** — это фрагмент объектного кода, который хранится в файле на диске и загружается для работы только по мере необходимости.
- Место в памяти, которое отводится для загрузки оверлеев, называется **оверлейной областью памяти**.

Загрузчики. Оверлей

- Представим, что имеется программа, состоящая из семи объектных файлов, которые называются F1, F2, ..., F7.



Загрузчики. Оверлей

- Допустим, имеющейся свободной памяти недостаточно для загрузки программы, скомпонованной обычным образом из всех объектных файлов.
- Есть возможность скомпоновать только первые четыре файла, а при большем количестве наступит переполнение памяти.
- Выйти из подобной ситуации можно, дав компоновщику команду создать оверлеи из файлов F5, F6 и F7.
- При каждом вызове функции, которая содержится в одном из этих файлов, администратор оверлейной загрузки (**программа, предоставляемая компоновщиком или редактором связей**) находит необходимый файл и помещает его в **оверлейную область памяти**, создавая условия для работы программы.
- Коды, которые получились при компиляции файлов F1 – F4, остаются **резидентными**

Загрузчики

- При этом, если обращение к модулю или библиотеке не происходит, они в оперативную память компьютера не загружаются – и память на хранение их кода не расходуется.
- Кроме того, если код функций какой-нибудь библиотеки используется несколькими программами (особенно это относится к системным библиотекам), то нет необходимости несколько раз загружать один и тот же фрагмент кода – можно по мере необходимости обращаться к уже загруженному в оперативную память фрагменту.
- Напротив, код библиотеки, который не используется в данный момент ни одной программой, может быть удален из оперативной памяти (при возникновении обращения к нему он будет заново подгружен в

Загрузчики

- В предельном случае вся исполняемая программа может представлять собой незначительный по объему базовый модуль, загружаемый в память, а остальные модули и библиотеки будут тогда вызываться и подгружаться по мере необходимости.
- Такая структура позволяет существенно повысить эффективность использования оперативной памяти.
- За обеспечение работы с динамически загружаемыми библиотеками и модулями отвечает **динамический загрузчик**.

Загрузчики

- Динамический загрузчик всегда есть в составе ОС.
- **В его задачи входит:**
 - следить за обращениями к динамически загружаемым библиотекам и модулям;
 - загружать в оперативную память используемые библиотеки;
 - следить за количеством обращений к каждой библиотеке;
 - освободить память, занимаемую неиспользуемыми библиотеками.

Загрузчики

- **Другая задача, решаемая динамическим загрузчиком** – обеспечение доступа исполняемой программы к используемым ею ресурсам пользовательского интерфейса.
- Задача динамического загрузчика – найти ресурс по имени, загрузить в оперативную память и предоставить программе доступ к нему, а при завершении использования – освободить память, занятую ресурсом.

Загрузчики

- **В современных системах программирования загрузчик, как правило, отсутствует – его роль выполняет ОС.**