Тема 11. Системы программирования

Системы программирования – комплекс инструментальных программных средств, представляющий сервисные возможности для разработки компьютерных программ.

В состав системного программного обеспечения входят:

трансляторы с языков высокого уровня;

средства редактирования, компоновки и загрузки программ;

макроассемблеры (машинно-ориентированные языки);

отладчики машинных программ.

Системы программирования содержат:

текстовый редактор (набор и редактирование текста программы);

загрузчик;

компилятор (компиляция или интерпретация исходного текста программы в машинный код с диагностикой синтаксических и логических ошибок);

отладчик (проверка и отладка программы);

диспетчер файлов (работа с файлами: сохранение, поиск, уничтожение и т.д.).

Ядро системы программирования составляет язык программирования, которые разделяют на языки низкого и высокого уровня.

Языки низкого уровня (машинно- ориентированные) позволяют создавать программы в машинных кодах. Программирование на таких языках требует высокой квалификации программиста и больших трудозатрат, однако полученные программы занимают меньше места в памяти и работают быстрее. Обычно с помощью этих языков разрабатываются системные программы и драйверы.

Программы на языках **высокого уровня** близки к естественному (английскому) языку и представляют набор заданных команд.

Наиболее известные системы программирования:

Фортран (FORmula TRANslating system − система трансляции формул) в настоящее время активно используемый при решении задач математической ориентации.

Бейсик (Beginner’s All-purpose Symbolic Instruction Code – универсальный символический код инструкций для начинающих) достаточно популярный, по числу пользователей, язык.

Си − широко используется при создании системного программного обеспечения.

Паскаль (Pascal – назван в честь ученого Блеза Паскаля) разработан для обучения программированию, однако популярен и среди профессионалов.

Джава (Java) – платформенно-независимый язык объективно-ориентированного программирования, широко используется при создания интерактивных веб-страниц.

По структуре, уровню формализации входного языка и целевому назначению различают системы программирования машинно-ориентированные и машинно-независимые. Машинно-ориентированные системы программирования имеют входной язык, наборы операторов и изобразительные средства которых существенно зависят от особенностей ЭВМ (внутреннего языка, структуры памяти и т.д.). Достоинствами машинно-ориентированных систем является высокое качество создаваемых программ, разработанных для конкретных аппаратных ресурсов, предсказуемость объектного кода. Недостатками таких систем является необходимо знания системы команд и особенности функционирования конкретного процессора, высокая трудоемкость процесса составления программ, низкая защищенность от ошибок, невозможность переноса на другие системы. Машинно-ориентированные системы по степени автоматического программирования подразделяются на классы:

1.Машинный язык. В таких системах программирования каждый тип компьютера имеет свой машинный язык. Некоторые семейства ЭВМ имеют единый машинный язык. Современные машинные языки по своим функциональным возможностям приближаются к алгоритмическим языкам программирования.

2.Система символического кодирования являются командным языком, в котором коды операций и адреса в машинных командах, представляющие собой последовательность двоичных или восьмеричных цифр, заменены символами (идентификаторами). Это позволяет существенно уменьшить число ошибок при составлении программ. Команды ЭВМ вместо истинных адресов содержат символические адреса. По результатам составленной программы определяется требуемое количество ячеек для хранения исходных промежуточных и результирующих значений. Назначение адресов, выполняемое отдельно от составления программы в символических адресах, может проводиться менее квалифицированным программистом или специальной программой.

3.Автокоды – система символического кодирования, расширенная посредством расширенного введения макрокоманд. В различных программах встречаются некоторые достаточно часто использующиеся командные последовательности, которые соответствуют определенным процедурам преобразования информации. Эффективная реализация таких процедур обеспечивается оформлением их в виде специальных макрокоманд и включением последних в язык программирования, доступный программисту. Макрокоманды переводятся в машинные команды двумя путями – расстановкой и генерированием. В постановочной системе содержатся «остовы» – серии команд, реализующие требуемую функцию, обозначенную макрокомандой. Макрокоманды обеспечивают передачу фактических

параметров, которые в процессе трансляции вставляются в «остов» программы, превращая её в реальную машинную программу. В системе с генерацией имеются специальные программы, анализирующие макрокоманду, которые определяют, какую функцию необходимо выполнить и формируют необходимую последовательность команд, реализующих данную функцию. Развитые автокоды получили название Ассемблеры.

4.Макрос. Предназначен для сокращения записи исходной программы. Компонент программного обеспечения, обеспечивающий функционирование макросов, называется макропроцессором. На макропроцессор поступает макросопределяющий и исходный текст. Реакция макропроцессора на вызов – выдача выходного текста. Макрос одинаково может работать, как с программами, так и с данными. Машинно-независимые системы программирования – это средство описания алгоритмов решения задач и информации, подлежащей обработке. Они удобны в использовании для широкого круга пользователей и не требуют от них знания особенностей организации функционирования ЭВМ. В таких системах программы, составляемые языках, имеющих название высокоуровневых языков программирования, представляют собой последовательности операторов, структурированные согласно правилам рассматривания языка (задачи, сегменты, блоки и т.д.).

Среди **машинно-независимых систем программирования** следует выделить:

1.Процедурно-ориентированные системы. Языки программирования таких система служат для записи алгоритмов, характерных для решения задач определенного класса. Эти языки, должны обеспечить программиста средствами, позволяющими коротко и четко формулировать задачу и получать результаты в требуемой форме. Процедурных языков очень много, например: Фортран, Алгол, Simula, Слэнг, Лисп, Снобол.

2.Проблемно-ориентированные системы в качестве входного языка используют язык программирования с проблемной ориентацией. С расширением областей применения вычислительной техники возникла необходимость формализовать представление постановки и решение новых классов задач. Необходимо было создать такие языки программирования, которые, используя в данной области обозначения и терминологию, позволили бы описывать требуемые алгоритмы решения для поставленных задач. Эти языки, ориентированные на решение определенных проблем, должны обеспечить программиста средствами, позволяющими коротко и четко формулировать задачу и получать результаты в требуемой форме. Программы, составленные на основе этих языков программирования, записаны в терминах решаемой задачи и реализуются выполнением соответствующих процедур.

3.Диалоговые языки. Программные средства, обеспечивающие оперативное взаимодействие оператора с компьютером. При внесении изменений в программу система программирования с помощью специальных таблиц устанавливает взаимосвязь структур исходной и объектной программ.

Это позволяет осуществить требуемые редакционные изменения в объектной программе.

4.Непроцедурные языки. Непроцедурные языки составляют группу языков, описывающих организацию данных, обрабатываемых по фиксированным алгоритмам (табличные языки и генераторы отчетов), и языков связи с операционными системами. Позволяя четко описывать как задачу, так и необходимые для её решения действия, таблицы решений дают возможность в наглядной форме определить, какие условия должны выполнятся, прежде чем переходить к какому-либо действию. Одна таблица решений, описывающая некоторую ситуацию, содержит все возможные блок-схемы реализаций алгоритмов решения. Табличные методы легко осваиваются специалистами любых профессий. Программы, составленные на табличном языке, удобно описывают сложные ситуации, возникающие при системном анализе.

В самом общем случае для создания программы на выбранном языке программирования нужно иметь следующие компоненты.

1.Текстовый редактор. Специализированные текстовые редакторы, ориентированные на конкретный язык программирования, необходимы для получения файла с исходным текстом программы, который содержит набор стандартных символов для записи алгоритма.

2.Исходный текст с помощью программы-компилятора переводится в машинный код. Исходный текст программы состоит, как правило, из нескольких модулей (файлов с исходными текстами). Каждый модуль компилируется в отдельный файл с объектным кодом, которые затем требуется объединить в одно целое. Кроме того, системы программирования, как правило, включают в себя библиотеки стандартных подпрограмм. Стандартные подпрограммы имеют единую форму обращения, что создает возможности автоматического включения таких подпрограмм в вызывающую программу и настройки их параметров.

3.Объектный код модулей и подключенные к нему стандартные функции обрабатывает специальная программа – **редактор связей.** Данная программа объединяет объектные коды с учетом требований операционной системы и формирует на выходе работоспособное приложение – исполнимый код для конкретной платформы. Исполнимый код это законченная программа, которую можно запустить на любом компьютер, где установлена операционная система, для которой эта программа создавалась.

4. В современных системах программирования имеется еще один компонент – **отладчик,** который позволяет анализировать работу программы во время ее исполнения. С его помощью можно последовательно выполнять отдельные операторы исходного текста последовательно, наблюдая при этом, как меняются значения различных переменных.

5. В последние несколько лет в программировании (особенно для операционной среды Windows) наметился так называемый **визуальный подход.** Этот процесс автоматизирован в средах быстрого проектирования. При

этом используются готовые визуальные компоненты, свойства и поведение которых настраиваются с помощью специальных редакторов. Таким образом, происходит переход от языков программирования системного уровня к языкам сценариев. Эти языки создавались для различных целей, что обусловило ряд фундаментальных различий между ним. Системные разрабатывались для построения структур данных и алгоритмов “с нуля”, начиная от таких примитивных элементов, как слово памяти компьютера. В отличие от этого, языки описания сценариев создавались для связывания готовых программ. Их применение подразумевает наличие достаточного ассортимента мощных компонентов, которые требуется только объединить друг с другом. Языки системного уровня используют строгий контроль типов данных, что помогает разработчикам приложении справляться со сложными задачами. Языки описания сценариев не используют понятие типа, что упрощает установление связей между компонентами, а также ускоряет разработку прикладных систем. Языки описания сценариев основаны на несколько другом наборе компромиссов, чем языки системного уровня. В них скорость исполнения и строгость контроля типов ставятся в шкале приоритетов на более низкое место, но зато выше цениться производительность труда программиста и повторное использование. Это соотношение ценностей оказывается все более обоснованным по мере того, как компьютеры становятся быстродействующими и менее дорогими, чего нельзя сказать о программистах. Языки системного программирования хорошо подходят для создания компонентов, где основная сложность заключена в реализации алгоритмов и структур данных, тогда как языки описания сценариев лучше приспособлены для построения приложении из готовых компонентов, где сложность состоит в налаживании межкомпонентных связей. Задачи последнего рода получают все большее распространение, так что роль языков описания сценариев будет возрастать.