ДИНАМИ́ЧЕСКОЕ РАСПРЕДЕЛЕ́НИЕ ПА́МЯТИ

Авторы: Л. Е. Карпов, А. В. Ермолович (актуализация)

ДИНАМИ́ЧЕСКОЕ РАСПРЕДЕЛЕ́НИЕ ПА́МЯТИ, способ распределения оперативной памяти ЭВМ, при котором она выделяется и освобождается *компилятором* или *операционной системой* по запросам программы в ходе её выполнения. Как правило, Д. р. п. выбирается в тех случаях, когда на стадии компиляции не удаётся определить положение объекта (напр., процедуры) в некоторой области памяти и/или его размер. Для осуществления Д. р. п. компилятор устанавливает соответствие между лексич. единицами (идентификаторы, метки, константы и др.) исполняемой программы и адресами, размерами, а также атрибутами областей памяти (совокупностями объединённых между собой элементов памяти) вычислит. системы, отведённых для хранения этих лексич. единиц при выполнении программы. Выбор области памяти и выделение в ней соответствующих фрагментов проводятся как для объектов данных, так и для выполняемых фрагментов программ – операторов, блоков, функций и процедур. Управление областями памяти выполняется операц. системой на основе использования технологий виртуальной памяти с сегментной, страничной или сегментно-страничной организацией. В динамич. области памяти могут располагаться как зоны, выделяемые пользователем (работающей программой), так и зоны, выделяемые компилятором автоматически, но и те и другие выделяются в процессе выполнения программы.

Существуют разл. методы Д. р. п., наиболее известными из которых являются стековый и метод произвольного распределения (распределение в «куче»). При стековом методе Д. р. п. на запросы ресурсов памяти выделяется некоторая область свободной оперативной памяти (стек), в которой проводится работа с фрагментами памяти по правилу «последний пришёл – первый вышел». Этот метод обычно выбирается для выделения памяти под локальные переменные блоков и процедур, а также под записи об активации процедур.

При произвольном методе Д. р. п. выделение и освобождение фрагментов памяти осуществляются по произвольным запросам без к.-л. системы. Произвольное Д. р. п. может производиться самим разработчиком программы (т. е. явно) или осуществляться автоматически (т. е. неявно). Д. р. п. для переменных, создаваемых по явному запросу, разработчик проводит, как правило, с помощью спец. операторов или библиотечных функций (напр., операторы ALLOCATE в языке PL/1, new в Pascal и Ci++, функция malloc в Ci). Эти операторы и функции, в свою очередь, могут использовать возможности операц. системы, а могут производить Д. р. п. самостоятельно, в рамках статически выделенного большого участка памяти.

Автоматическое Д. р. п. осуществляется если в программе используются такие операции над данными, которые требуют перераспределения памяти, а сами операторы перераспределения в программе отсутствуют [напр., операция слияния значений (конкатенации) строковых переменных в языке Basic или в языке Object Pascal]. Также автоматически проводится управление сегментами и страницами памяти в операц. системах, которые для этого могут пользоваться аппаратной поддержкой. При применении метода произвольного Д. р. п. возникает серьёзная проблема освобождения памяти и повторного использования освобождённых фрагментов (т. к. моменты возврата памяти не всегда очевидны, а порядок возврата памяти непредсказуем), которая имеет разл.

решения в зависимости от метода выделения участков памяти. Напр., производится периодическая «сборка мусора» (англ. Garbage Collection) – выделенная ранее память пересматривается с целью обнаружения неиспользуемых фрагментов, а высвобожденная память передаётся для повторного использования. Однако этот метод требует либо явной поддержки со стороны языка программирования (напр., Java), либо специализированного оборудования с тегированием указателей в памяти (напр., контекстная защита в семействе ЭВМ «Эльбрус»). Произвольное Д. р. п. также чревато фрагментацией доступной ОП, приводящей к невозможности выделения фрагмента памяти запрашиваемого размера несмотря на превосходящий его объём свободной памяти в системе ввиду раздробленности этого объёма на множество составляющих меньшего, нежели запрашиваемый, размера. Проблему можно решить при помощи компактировки занимаемой памяти, что, однако, также требует явной поддержки со стороны языка программирования либо специализированного оборудования. Аналогичная проблема в терминах физической, а не математической памяти решается за счёт использования механизма виртуальной памяти либо сегментации (ныне практически не используется).

Дисциплина выделения памяти на стеке в некоторой мере снимает проблему возврата памяти, выполняемого автоматически при возврате из очередной процедуры, но порождает новую проблему конкурентного распределения памяти в куче и на стеке. Проблема многократно усугубляется в многопоточных приложениях, оперирующих единой кучей и множеством стеков одновременно. Для языков с явной параллельностью исполнения процедур применение единого линейного стека алгоритмически невозможно; превращаясь в дерево, стек вызовов и локальных переменных фактически вырождается в кучу с дисциплиной автоматического освобождения памяти.

Литература

Лит.: Таненбаум А. Современные операционные системы. 2-е изд. СПб., 2002;Гордеев А. В., Молчанов А. Ю. Системное программное обеспечение. СПб., 2003.