Тема 7. УПРАВЛЕНИЕ ПАМЯТЬЮ

Функции ОС по управлению памятью

Под памятью (**memory**) здесь подразумевается *оперативная память* компьютера. В отличие от *внешней памятью* (**storage**) *оперативная память* (**memory**) для своей работы требует наличие постоянного электропитания.

Процессор может выполнять инструкции только в том случае, если они находятся в оперативной памяти. Память распределяется как между модулями прикладных программ, так и между модулями самой операционной системы.

Часть операционной системы, отвечающая за управление памятью называется ***модулем управления памятью*** или ***менеджером памяти***.

Функциями ОС по управлению памятью однозадачных ОС являются:

- выделение памяти для процесса пользователя при его запуске и освобождение этой памяти при завершении процесса;
- обеспечение настройки запускаемой программы на выделенные адреса памяти;
- управление выделенными областями памяти по запросам программы пользователя (например, освобождение части памяти перед запуском порожденного процесса).

Функциями ОС по управлению памятью в мультипрограммной системе являются:

- отслеживание свободной и занятой памяти;
- выделение памяти процессам и освобождение памяти по завершении процессов;
- вытеснение кодов и данных процессов из оперативной памяти на диск (полное или частичное), когда размеры основной памяти не достаточны для размещения в ней всех процессов, и возвращение их в оперативную память, когда в ней освобождается место;
- настройка адресов программы на конкретную область физической памяти;
- изоляция памяти процессов, исключающая случайное или намеренное несанкционированное обращение одного процесса к областям памяти, занимаемым другим процессом;
- предоставление процессам возможности обмена данными через общие области памяти.

Помимо выделения памяти при создании процессов, ОС управляет **динамическим распределением памяти (**обработкой запросов приложений на выделение им дополнительной памяти во время их выполнения).

Выделение памяти случайной длины в случайные моменты времени из общего пула памяти приводит к фрагментации и, вследствие этого, к

неэффективному ее использованию. Дефрагментация памяти тоже является функцией операционной системы.

Алгоритмы распределения памяти разделены на два класса: алгоритмы, в которых используется перемещение сегментов процессов между оперативной памятью и диском, и алгоритмы, в которых внешняя память не привлекается.

При распределении памяти фиксированными разделами память разбивается на области фиксированной величины (разделы) во время запуска системы. После этого границы разделов не изменяются.

Каждый новый процесс, поступивший на выполнение, помещается либо в ***общую очередь,*** либо в ***очередь к конкретному разделу***.

При распределении памяти динамическими разделами каждому новому процессу выделяется вся необходимая ему память (при ее наличии). Если памяти для выполнения процесса недостаточно, то он не создается. После завершения выполнения процесса вся занятая им память освобождается.

Во время работы оперативная память представляет собой случайную последовательность занятых и свободных участков произвольного размера.

Операционная система выполняет следующие действия по управления памятью:

- Ведение таблиц, содержащих адреса и размеры свободных и занятых областей.
- Выделение памяти для нового процесса. Для этого в таблице свободных областей находится раздел, размер которого достаточен для размещения кодов и данных нового процесса. Загрузка программы и настройка адресов в выделенной области памяти.
- Корректировка таблиц свободных и занятых областей после загрузки и выгрузки программы.

Для поиска подходящего по размеру участка памяти используются следующие алгоритмы:

Алгоритм первого подходящего участка – последовательно просматривает список свободных разделов до тех пор, пока не будет найден подходящий (достаточный для размещения процесса) участок. Является самым быстрым алгоритмом.

Алгоритм следующего подходящего участка − в отличии от алгоритма первого подходящего участка поиск ведется не с начала области памяти, а от последнего найденного подходящего участка. Производительность немного хуже предыдущего алгоритма.

Алгоритм самого подходящего участка выбирает наименьший из подходящих по размеру, размеру участок. Недостатки метода:

− низкая скорость, т.к. поиск ведется по всему списку;

− сильная фрагментация, из-за большого количества маленьких свободных участков.

Алгоритм самого неподходящего участка − выбирает наибольший по размеру, подходящий свободный участок. Использованием данного алгоритма достигается низкая фрагментации памяти, однако работает он медленнее других алгоритмов.

Алгоритм быстрого подходящего участка использует отдельные списки для нескольких часто запрашиваемых размеров. Алгоритм работает очень быстро. Однако при освобождении памяти (для устранения фрагментации) необходимо слияние неиспользованных областей с последующей сортировкой списков, что является очень затратной операцией.

При распределение памяти перемещаемыми разделами, для борьбы с фрагментацией все занятые участи перемещаются в сторону младших или старших адресов так, чтобы всея свободная память образовала единую свободную область. Такая операция называется ***уплотнением*** или ***сжатием памяти***.

Виртуальная память (virtual memory) — метод увеличения объема доступной программе памяти, за счет использования дисковой памяти. Для программы виртуальная память выглядит как доступное и непрерывное адресное пространство вне зависимости от наличия у компьютера соответствующего объёма оперативной памяти. Управление виртуальным адресным пространством выполняет операционная система.

В основе виртуальной памяти лежит идея, что *у каждой программы имеется свое собственное адресное пространство, которое разбивается на участки, называемые* ***страницами***. Каждая страница представляет собой непрерывный диапазон адресов. Эти страницы отображаются на физическую память, но для запуска программы присутствие в памяти всех страниц не обязательно. Если программа обращается к части адресного пространства, которая находится в физической памяти, аппаратное обеспечение осуществляет необходимые действия на лету. Если программа ссылается на часть своего адресного пространства, которое не находится в физической памяти, операционная система предупреждается о том, что необходимо получить недостающую часть и повторно выполнить потерпевшую неудачу команду. Пока программа ждет считывания какой-либо собственной части, центральный процессор может быть отдан другому процессу.

Существует два основных способа организации виртуальной памяти, которые базируются на использовании виртуальных адресов, аппаратно преобразуемых в физические при выполнении команды – ***сегментная и страничная (paging) организация памяти***.

При страничном распределение памяти виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые ***виртуальными страницами*** (virtual pages). Вся оперативная память машины также делится на части такого же размера, называемые ***физическими страницами*** (блоками, кадрами). Размер страниц выбирается равным 2^n (512, 1024, 4096 байт и т. д.).

При страничном выделении памяти адресное пространство непрерывно для всех частей процесса. Однако для некоторых процессов может быть оптимальным наличие нескольких отдельных виртуальных адресных пространств (для кода, данных или стека). Независимые виртуальные адресные пространства называются **сегментами.**

Каждый сегмент состоит из линейной последовательности адресов начинающихся с нуля. Длина каждого сегмента устанавливается исходя из текущих потребностей, и может изменяться в процессе выполнения программы.

При сегментно-страничное распределении памяти виртуальное адресное пространство процесса разделено на сегменты. Перемещение данных между памятью и диском осуществляется страницами. Для этого каждый виртуальный сегмент и физическая память делятся на страницы равного размера

Память (memory, storage) вычислительной машины представляет собой иерархию запоминающих устройств (ЗУ), различающихся средним временем доступа к данным, объемом и стоимостью хранения одного бита.

Регистры процессора используются в качестве ***сверхоперативной памяти*** для хранения небольших объемов информации и имеют очень небольшое время доступа (несколько наносекунд). Общий объем регистров может составлять десятки или сотни байт.

Кэш-память − память статического типа, с быстродействием, превышающим быстродействие оперативной памяти. Как правило, в компьютерах используется два уровня кэш-памяти (Level 1 и Level 2). Объем кэш-памяти 1-го уровня составляет от десятков до сотен Кбайт. Объем кэш - памяти 2-го уровня составляет от сотен Кбайт до нескольких Мбайт.

Кэш-память предназначена для выравнивания скорости работы регистров процессора и оперативной памяти. Для этого в кэш-память из *оперативной памяти* подкачиваются *команды и данные*, которые могут быть востребованы процессором в ближайшем будущем. Выигрыш в быстродействии значительно больше затрат, связанных с перезаписью программ из оперативной памяти в кэш-память и обратно.

Оперативная память (основная или первичная) (ОП, ОЗУ) (read/write memory, main memory, main stor, main storage, primary memory) – запоминающее устройство произвольной выборки (RAM), служащее основным хранилищем информации выполняющихся программ и данных.

Внешняя память служит для хранения данных и программ, которые могут быть загружены в оперативную память для выполнения. Процессор не имеет непосредственного доступа к внешней памяти.

Кэширование — способ совместного функционирования различных типов запоминающих устройств позволяющий уменьшать среднее время доступа к данным и экономить более дорогую быстродействующую память.

Работа с каш-памятью обеспечивается ОС и не требует вмешательства программиста.

Содержимое кэш-памяти представляет собой совокупность записей, имеющих следующие поля: адрес данных в основной памяти, сами данные, управляющая информация.

При каждом обращении к основной памяти по физическому адресу просматривается содержимое кэш-памяти с целью определения, не находятся ли там нужные данные. Так как кэш-память ***не является адресуемой***, поиск нужных данных осуществляется по содержимому, а по взятому из запроса значению поля адреса в оперативной памяти.

Чтение данных происходит следующим образом:

– если данные обнаруживаются в кэш-памяти, т.е. произошло кэш-попадание (cache-hit), то они считываются из неё и результат передаётся источнику запроса;

– если нужные данные отсутствуют в кэш-памяти, т.е. произошёл кэш-промах (cache-miss), то они считываются из ОП, передаются источнику запроса и одновременно с этим копируются в кэш-память.

При кэшировании данных из оперативной памяти широко используются две основные схемы размещения:

– *случайное размещение;*

– *детерминированное размещение.*

При ***случайном размещении*** элементы из оперативной памяти размещаются в произвольном месте кэш-памяти. При каждом запросе к оперативной памяти выполняется поиск в кэше, а критерием поиска выступает адрес оперативной памяти из запроса.

Так как метод простого перебора для поиска данных в кэше работает очень медленно, то используется **ассоциативный поиск**, при котором сравнение выполняется не последовательно, а параллельно по всем записям. Аппаратная реализация ассоциативного поиска достаточно дорога.

В кэшах со ***случайным размещением*** вытеснение старых данных происходит только в случае заполнения всей кэш-памяти.

Алгоритмы выбора данных для выгрузки аналогичны алгоритмам, использующимся при замещении страниц.

При использовании детерминированного алгоритма размещения каждый элемент оперативной памяти всегда отображается в одно и то же место кэш-памяти. Кэш-память разделена на строки, каждая из которых имеет свой номер и предназначена для хранения записи об одном или нескольких элементах данных.

Между номерами строк кэш-памяти и адресами оперативной памяти устанавливается соответствие «один ко многим»: одному номеру строки соответствует несколько адресов оперативной памяти.

В качестве отображающей функции могут использоваться нескольких разрядов из адреса оперативной памяти, которые интерпретируются как номер строки кэш-памяти (такое отображение называется ***прямым***).

Так как в найденной строке могут находиться данные из любой ячейки оперативной памяти с совпадающим младшим адресом, то каждая строка кэш-памяти дополняется тегом, содержащим старшую часть адреса данных в оперативной памяти.

При совпадении тега с соответствующей частью адреса из запроса констатируется кэш-попадание.

В случае кэш-промаха, имеющиеся в кэше данные вытесняются, а в кэш записываются запрашиваемые данные.

В современных процессорах кэш-память строится на основе сочетания двух подходов, как компромисс между интеллектуальностью алгоритмов замещения в кэше со случайным отображением низкой стоимостью кэша с прямым отображением. При ***смешанном*** подходе адрес оперативной памяти отображается на номер группы адресов в кэше.

Поиск в кэше осуществляется сначала по номеру группы, полученному из адреса оперативной памяти из запроса, а затем в пределах группы путём ассоциативного поиска записей в группе на предмет совпадения старших частей адресов оперативной памяти.

При кэш-промахе данные копируются по любому свободному адресу из заданной группы. Если свободных адресов в группе нет, то выполняется вытеснение данных.