# ПРАКТИЧЕСКАЯ РАБОТА №4

# «МОДЕЛИРОВАНИЕ АЛГОРИТМОВ УПРАВЛЕНИЯ ЛОКАЛЬНЫМ РЕСУРСОМ «ПАМЯТЬ» В ОПЕРАЦИОННЫХ СИСТЕМАХ»

**Цель работы:** изучить основные алгоритмы управления памятью в операционных системах, разработать программную модель управления памятью в соответствии с вариантом задания.

## 1.1 Теоретические сведения

### Управление памятью

Все методы управления памятью могут быть разделены на два класса: методы, которые используют перемещение процессов между оперативной памятью и диском, и методы, которые не делают этого. Ниже рассмотрены основные алгоритмы управления памятью.

### Методы распределения памяти без использования дискового пространства

*Распределение памяти фиксированными разделами*

Самым простым способом управления оперативной памятью является разделение ее на несколько разделов фиксированной величины. Это может быть выполнено вручную оператором во время старта системы или во время ее генерации. Очередная задача, поступившая на выполнение, помещается либо в общую очередь, либо в очередь к некоторому разделу. Подсистема управления памятью в этом случае выполняет следующие задачи: во-первых, сравнивая размер программы, поступившей на выполнение, и свободных разделов, выбирает подходящий раздел и, во-вторых, осуществляет загрузку программы и настройку адресов.

Достоинства метода: простота реализации.

Недостатки метода:

- так как в каждом разделе может выполняться только одна программа, то уровень мультипрограммирования ограничен числом разделов;
- даже если программа имеет небольшой объем, она будет занимать весь раздел;
- даже если объем оперативной памяти машины позволяет выполнить некоторую программу, разбиение памяти на разделы не позволяет сделать этого.

*Распределение памяти разделами переменной величины*

В этом случае память машины не делится заранее на разделы. Сначала вся память свободна. Каждой вновь поступающей задаче выделяется необходимая ей память. Если достаточный объем памяти отсутствует, то задача не принимается на выполнение и стоит в очереди. После завершения задачи память освобождается, и на это место может быть загружена другая задача. Таким образом, в произвольный момент времени оперативная память представляет собой случайную последовательность занятых и свободных участков (разделов) произвольного размера.

Задачами ОС в этом случае является:

- ведение таблиц свободных и занятых областей;
- при поступлении новой задачи, просмотр таблицы свободных областей и выбор раздела, размер которого достаточен для размещения поступившей задачи;

- загрузка задачи в выделенный раздел и корректировка таблиц свободных и занятых областей;
- после завершения задачи корректировка таблиц свободных и занятых областей.

Достоинства метода: обладает гораздо большей гибкостью чем предыдущий.

Недостаток метода: фрагментация памяти (т.е. наличие большого числа несмежных участков свободной памяти очень маленького размера (фрагментов), таких что ни одна из вновь поступающих программ не может поместиться ни в одном из участков, хотя суммарный объем фрагментов может составить величину, превышающую требуемый объем памяти).

*Перемещаемые разделы*

Одним из методов борьбы с фрагментацией является перемещение всех занятых участков в сторону старших либо в сторону младших адресов, так, чтобы вся свободная память образовывала единую свободную область. В дополнение к функциям, которые выполняет ОС при распределении памяти переменными разделами, в данном случае она должна еще время от времени копировать содержимое разделов из одного места памяти в другое, корректируя таблицы свободных и занятых областей. Эта процедура называется "сжатием". Сжатие может выполняться либо при каждом завершении задачи, либо только тогда, когда для вновь поступившей задачи нет свободного раздела достаточного размера. В первом случае требуется меньше вычислительной работы при корректировке таблиц, а во втором - реже выполняется процедура сжатия.

Достоинства метода: более эффективное использование памяти

Недостатки метода: процедура сжатия может требовать значительного времени, что часто перевешивает преимущества данного метода.

**Методы распределения памяти с использованием дискового пространства**

*Страничное распределение*

На рисунке 1.1 показана схема страничного распределения памяти. Виртуальное адресное пространство каждого процесса делится на части одинакового, фиксированного для данной системы размера, называемые виртуальными страницами. В общем случае размер виртуального адресного пространства не является кратным размеру страницы, поэтому последняя страница каждого процесса дополняется фиктивной областью.

Вся оперативная память машины также делится на части такого же размера, называемые физическими страницами (или блоками).

При загрузке процесса часть его виртуальных страниц помещается в оперативную память, а остальные - на диск. При загрузке операционная система создает для каждого процесса информационную структуру - таблицу страниц, в которой устанавливается соответствие между номерами виртуальных и физических страниц для страниц, загруженных в оперативную память, или делается отметка о том, что виртуальная страница выгружена на диск. Кроме того, в таблице страниц содержится управляющая информация, такая как признак модификации страницы, признак невыгружаемости (выгрузка некоторых страниц может быть запрещена), признак обращения к странице (используется для подсчета числа обращений за определенный период времени) и другие данные, формируемые и используемые механизмом виртуальной памяти.

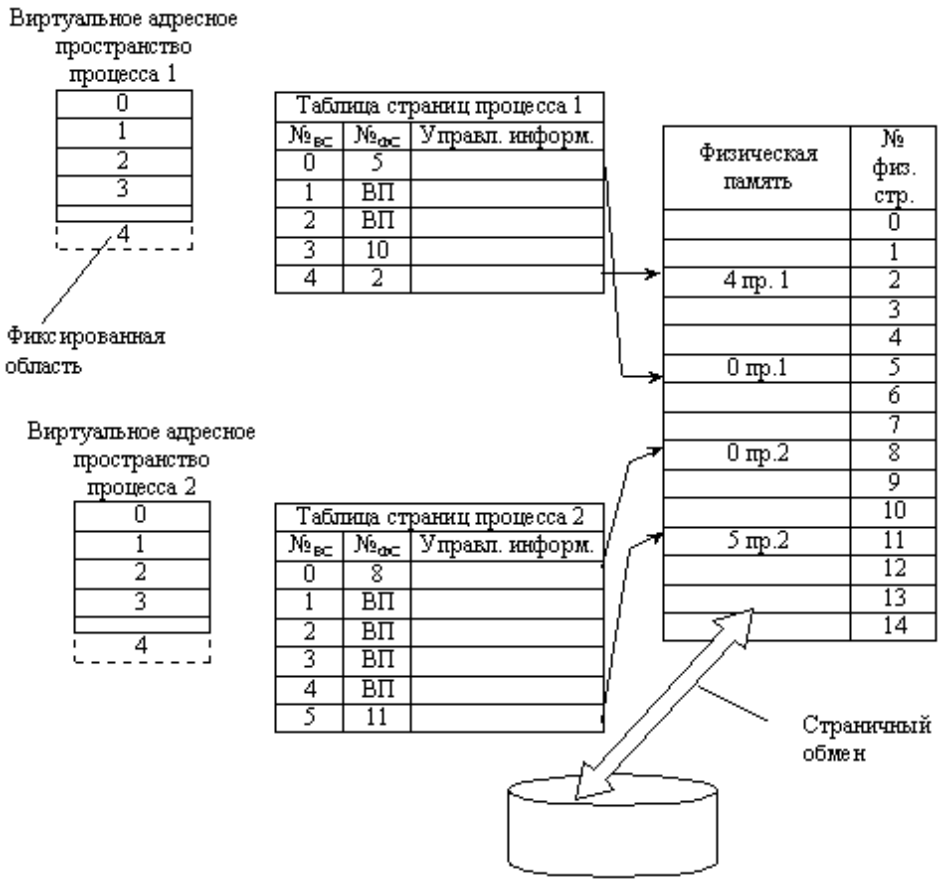


Рис. 1.1. Страничное распределение памяти

При каждом обращении к памяти происходит чтение из таблицы страниц информации о виртуальной странице, к которой произошло обращение. Если данная виртуальная страница находится в оперативной памяти, то выполняется преобразование виртуального адреса в физический. Если же нужная виртуальная страница в данный момент выгружена на диск, то происходит так называемое страничное прерывание. Выполняющийся процесс переводится в состояние ожидания, и активизируется другой процесс из очереди готовых. Параллельно программа обработки страничного прерывания находит на диске требуемую виртуальную страницу и пытается загрузить ее в оперативную память. Если в памяти имеется свободная физическая страница, то загрузка выполняется немедленно, если же свободных страниц нет, то решается вопрос, какую страницу следует выгрузить из оперативной памяти.

В данной ситуации может быть использовано много разных критериев выбора, наиболее популярные из них следующие: дольше всего не использовавшаяся страница; первая попавшаяся страница; страница, к которой в последнее время было меньше всего обращений.

Страничное распределение памяти может быть реализовано в упрощенном варианте, без выгрузки страниц на диск. В этом случае все виртуальные страницы всех процессов постоянно находятся в оперативной памяти. Такой вариант страничной организации хотя и не предоставляет пользователю виртуальной памяти, но почти исключает фрагментацию за счет того, что программа может загружаться в несмежные области, а также того, что при загрузке виртуальных страниц никогда не образуется остатков.

*Сегментное распределение*

При страничной организации виртуальное адресное пространство процесса делится механически на равные части. Это не позволяет дифференцировать способы доступа к разным частям программы (сегментам), а это свойство часто бывает очень полезным.

Кроме того, разбиение программы на "осмысленные" части делает принципиально возможным разделение одного сегмента несколькими процессами.

При сегментном распределении памяти (рис. 1.2) виртуальное адресное пространство процесса делится на сегменты, размер которых определяется программистом с учетом смыслового значения содержащейся в них информации. Отдельный сегмент может представлять собой подпрограмму, массив данных и т.п. Иногда сегментация программы выполняется по умолчанию компилятором.

При загрузке процесса часть сегментов помещается в оперативную память (при этом для каждого из этих сегментов операционная система подыскивает подходящий участок свободной памяти), а часть сегментов размещается в дисковой памяти. Сегменты одной программы могут занимать в оперативной памяти несмежные участки. Во время загрузки система создает таблицу сегментов процесса (аналогичную таблице страниц), в которой для каждого сегмента указывается начальный физический адрес сегмента в оперативной памяти, размер сегмента, правила доступа, признак модификации, признак обращения к данному сегменту за последний интервал времени и некоторая другая информация. Если виртуальные адресные пространства нескольких процессов включают один и тот же сегмент, то в таблицах сегментов этих процессов делаются ссылки на один и тот же участок оперативной памяти, в который данный сегмент загружается в единственном экземпляре.

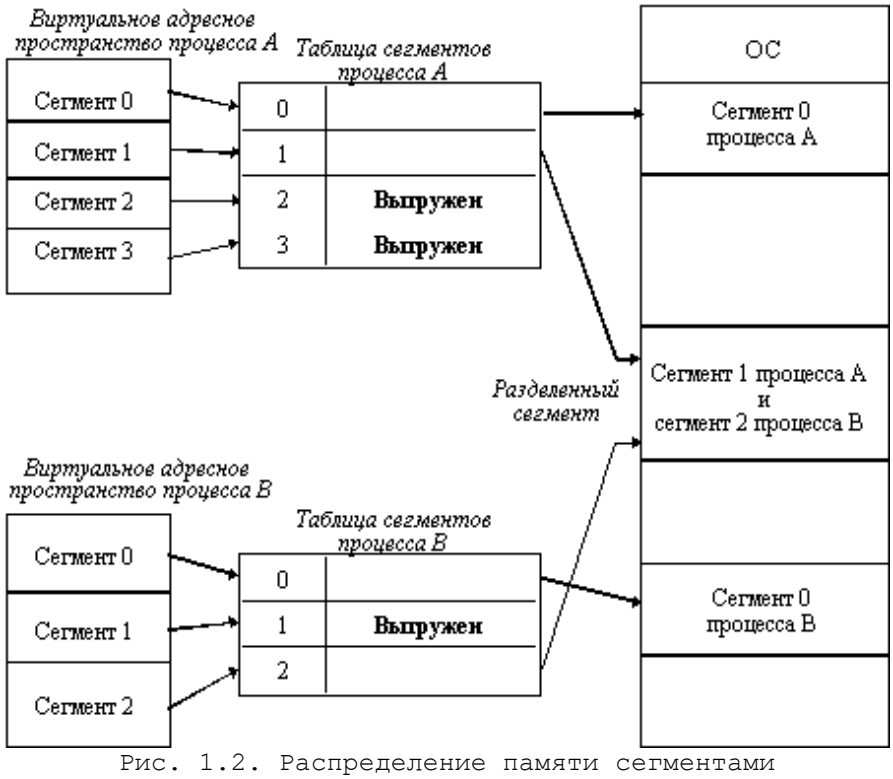


Рис. 1.2. Распределение памяти сегментами

Система с сегментной организацией функционирует аналогично системе со страничной организацией: время от времени происходят прерывания, связанные с отсутствием нужных сегментов в памяти, при необходимости освобождения памяти некоторые сегменты выгружаются. Кроме того, при обращении к памяти проверяется, разрешен ли доступ требуемого типа к данному сегменту.

Недостатком данного метода распределения памяти является фрагментация на уровне сегментов и более медленное по сравнению со страничной организацией преобразование адреса.

*Странично-сегментное распределение*

Данный метод представляет собой комбинацию страничного и сегментного распределения памяти и, вследствие этого, сочетает в себе достоинства обоих подходов. Виртуальное пространство процесса делится на сегменты, а каждый сегмент в свою очередь делится на виртуальные страницы, которые нумеруются в пределах сегмента. Оперативная память делится на физические страницы. Загрузка процесса выполняется операционной системой постранично, при этом часть страниц размещается в оперативной памяти, а часть на диске. Для каждого сегмента создается своя таблица страниц, структура которой полностью совпадает со структурой таблицы страниц, используемой при страничном распределении. Для каждого процесса создается таблица сегментов, в которой указываются адреса таблиц страниц для всех сегментов данного процесса. Адрес таблицы сегментов загружается в специальный регистр процессора, когда активизируется соответствующий процесс.

*Свопинг*

Разновидностью виртуальной памяти является свопинг.

Анализ загрузки процессора в зависимости от числа одновременно выполняемых процессов показал, что для загрузки процессора на 90% достаточно всего трех счетных задач. Однако для того, чтобы обеспечить такую же загрузку интерактивными задачами, выполняющими интенсивный ввод-вывод, потребуются десятки таких задач. В этих условиях был предложен метод организации вычислительного процесса, называемый свопингом. В соответствии с этим методом некоторые процессы (обычно находящиеся в состоянии ожидания) временно выгружаются на диск. Планировщик операционной системы не исключает их из своего рассмотрения, и при наступлении условий активизации некоторого процесса, находящегося в области свопинга на диске, этот процесс перемещается в оперативную память. Если свободного места в оперативной памяти не хватает, то выгружается другой процесс.

При свопинге, в отличие от рассмотренных ранее методов реализации виртуальной памяти, процесс перемещается между памятью и диском целиком.

**Задание на практическую работу**

Разработать программу, моделирующую один из алгоритмов управления памятью в соответствии с вариантом задания. При моделировании считать что:

- объем моделируемой «памяти» составляет 64К;
- поступаемые на выполнение задачи содержатся в отдельных структурах, содержащих количество килобайт либо страниц (для страничной организации памяти), необходимых для выполнения программы (моделируем размер задачи);
- размер задачи в диапазоне от 1 до 8 кбайт (кратно 1 кбайту), либо от 1 до 8 страниц для страничной организации памяти.
- для простоты моделирования разрешено заранее сформировать список доступных разделов в Варианте 1 и 2.
- Провести сравнение производительности реализованного алгоритма управления памятью относительно алгоритма управления памятью с постоянными разделами на примере n-количества задача (100 и больше). Результат сравнения представить в процентах.

Программа должна иметь возможность просмотра состояния моделируемой «памяти».

## Варианты заданий

| Вариант | Алгоритм управления памятью |
|---|---|
| 1 | Распределение памяти перемещаемыми разделами |
| 2 | Распределение памяти разделами переменной величины |
| 3 | Страничное распределение памяти |