**1. Что такое физическая память?**

Физическая память – это аппаратный компонент компьютера, представленный модулями оперативной памяти (RAM). Она используется для временного хранения данных и инструкций, необходимых процессору для выполнения задач. Доступ к физической памяти осуществляется с высокой скоростью, и объем физической памяти ограничен установленными модулями RAM.

**2. Что такое логическая память?**

Логическая память – это модель памяти, представленная процессу в виде непрерывного адресного пространства. Она абстрагирует физическую память, позволяя приложениям работать с памятью без необходимости знать ее физическое расположение. Адреса в логической памяти преобразуются в физические адреса через механизмы операционной системы.

**3. Что такое виртуальная память?**

Виртуальная память – это механизм, который расширяет объем доступной памяти за счет использования пространства на диске (например, файла подкачки). Она создает иллюзию большого непрерывного адресного пространства, независимо от ограничений физической памяти. Виртуальная память используется для изоляции процессов, повышения производительности и обеспечения мультизадачности.

**4. Как устроена виртуальная память?**

Виртуальная память организована следующим образом:

1. **Адресное пространство**: Процесс видит виртуальное адресное пространство, которое делится на страницы фиксированного размера.
2. **Страницы и фреймы**:
   * Виртуальная память делится на **страницы**.
   * Физическая память делится на **фреймы** того же размера.
3. **Таблица страниц**: ОС поддерживает таблицу страниц, сопоставляющую виртуальные адреса физическим.
4. **Механизм преобразования**:
   * **MMU (Memory Management Unit)** выполняет преобразование виртуальных адресов в физические.
   * Если нужная страница отсутствует в памяти (промах страницы), ОС загружает ее из файла подкачки.
5. **Файл подкачки**: Область на диске, используемая для хранения данных страниц, не находящихся в физической памяти.

**5. Перечислите и поясните назначение секций памяти процесса.**

Основные секции памяти процесса:

1. **Секция кода**:
   * Содержит исполняемый машинный код программы.
   * Обычно доступна только для чтения.
2. **Секция данных**:
   * Хранит глобальные и статические переменные, инициализированные в программе.
3. **Секция BSS**:
   * Хранит глобальные и статические переменные, неинициализированные при запуске.
   * Значения таких переменных автоматически равны нулю.
4. **Куча (Heap)**:
   * Используется для динамического выделения памяти в runtime.
   * Управляется вручную разработчиком (например, через malloc/free в C или new/delete в C++).
5. **Стек (Stack)**:
   * Используется для хранения локальных переменных, адресов возврата и параметров функций.
   * Управляется автоматически: выделение и освобождение происходит при вызовах функций.
6. **Виртуальная память (Mapping)**:
   * Секция, используемая для отображения файлов или ресурсов (например, библиотек).

**6. Какие API входят в WinAPI для работы с памятью? В POSIX?**

**WinAPI:**

* **Выделение/освобождение памяти**:
  + VirtualAlloc, VirtualFree
* **Работа с файлами памяти**:
  + MapViewOfFile, UnmapViewOfFile
* **Управление страницами памяти**:
  + VirtualProtect, VirtualQuery
* **Динамическая память**:
  + HeapAlloc, HeapFree
  + Работа с кучами: HeapCreate, HeapDestroy
* **Кэш и буферы**:
  + FlushViewOfFile

**POSIX:**

* **Выделение памяти**:
  + mmap, munmap
* **Изменение атрибутов памяти**:
  + mprotect
* **Сброс изменений в файл**:
  + msync
* **Динамическая память**:
  + malloc, free, realloc
* **Работа с анонимной памятью**:
  + Использование MAP\_ANONYMOUS с mmap.

**7. Когда лучше использовать виртуальную память напрямую?**

Виртуальную память напрямую следует использовать в следующих случаях:

1. **Работа с большими объемами данных**:
   * Когда требуется управлять памятью с точным контролем над выделением страниц.
2. **Производительность и контроль**:
   * Если важно настроить атрибуты страниц (например, сделать их только для чтения или запрещенными для выполнения).
3. **Память для отображения файлов**:
   * Для отображения содержимого файла в адресное пространство процесса (mmap, MapViewOfFile).
4. **Изоляция и безопасность**:
   * Для создания независимых регионов памяти, которые не смешиваются с памятью стандартных аллокаторов.

**8. Когда лучше использовать кучи?**

Кучи удобнее использовать в следующих случаях:

1. **Динамическое выделение небольших блоков памяти**:
   * Для работы с объектами, структурами или массивами, которые создаются и удаляются во время выполнения программы.
2. **Упрощение управления памятью**:
   * Использование библиотечных функций для автоматического управления памятью (например, malloc в C или new в C++).
3. **Стандартные задачи**:
   * Для большинства приложений, где требуется управлять динамическими структурами данных (например, списки, деревья, хеш-таблицы).