**14. Понятие «виртуальная память»**

**Виртуальная память** — это механизм управления памятью в операционной системе, позволяющий создавать иллюзию большого объема памяти для приложения. Она использует комбинацию физической оперативной памяти (RAM) и места на диске (файл подкачки). Это позволяет запускать программы, требующие больше памяти, чем есть в наличии.

**15. Понятие «свопинг»**

**Свопинг** — это процесс перемещения данных из оперативной памяти на диск и обратно, чтобы освободить RAM для более приоритетных задач. Данные, которые временно не используются, сохраняются в файл подкачки.

**16. Понятие «страничная память»**

**Страничная память** — это способ организации виртуальной памяти, при котором память делится на небольшие блоки фиксированного размера, называемые страницами. Это упрощает управление памятью, предотвращает фрагментацию и облегчает свопинг.

**17. Понятие MMU**

**MMU (Memory Management Unit)** — это аппаратный компонент процессора, управляющий преобразованием виртуальных адресов в физические. Он обеспечивает изоляцию процессов и поддержку страничной памяти.

**18. Понятие TLB**

**TLB (Translation Lookaside Buffer)** — это небольшой кеш в составе MMU, хранящий последние преобразования виртуальных адресов в физические. Это ускоряет доступ к памяти за счет уменьшения количества обращений к таблице страниц.

**19. Информация в строке таблицы страниц**

Строка таблицы страниц содержит:

* Физический адрес страницы.
* Флаги (например, права доступа, присутствие в памяти, модифицирована ли страница).
* Данные о состоянии страницы (например, заблокирована, находится в кеше).

**20. Принцип применения хэш-таблиц**

**Хэш-таблицы** используются для быстрого поиска и доступа к данным. Ключ преобразуется в индекс с помощью хэш-функции. В случае коллизий применяются методы разрешения (цепочки, открытая адресация).

**21. Применение «инвертированной таблицы страниц»**

**Инвертированная таблица страниц** содержит одну запись на каждую физическую страницу, указывая, какой виртуальный адрес ей соответствует. Это экономит память для больших адресных пространств, но увеличивает время поиска.

**22. Windows: области адресного пространства (от младших к старшим адресам)**

1. **Код программы** — содержит инструкции исполняемого файла.
2. **Данные программы** — глобальные переменные.
3. **Heap** — область динамически выделяемой памяти.
4. **Stack** — область стека, используется для локальных переменных и вызовов функций.
5. **Kernel Space** — область, зарезервированная для операционной системы.

**23. Windows: стандартный начальный размер области heap**

Стандартный начальный размер heap составляет **1 МБ**.

**24. Windows: стандартный размер области памяти stack**

Стандартный размер stack составляет **1 МБ** (настраиваемо при компиляции).

**25. Windows: «рабочее множество» и его управление**

**Рабочее множество** — это подмножество страниц процесса, которые находятся в RAM. ОС управляет рабочим множеством, добавляя страницы, когда они запрашиваются, и вытесняя их в файл подкачки при превышении лимитов. Для управления используются функции OS API, например, SetProcessWorkingSetSize.

**26. Windows: понятие «heap» и его устройство**

* **Heap** — это область динамически выделяемой памяти процесса.
* **Heap процесса** — основной heap, создаваемый при запуске процесса.
* **Пользовательская heap** — дополнительные heaps, создаваемые с помощью API, например, HeapCreate. Heap организуется как дерево свободных и занятых блоков памяти с минимизацией фрагментации.

**27. Linux: области адресного пространства (от младших к старшим адресам)**

1. **Текст** — содержит код программы.
2. **Данные** — глобальные и статические переменные.
3. **Heap** — область для динамической памяти.
4. **Stack** — стек вызовов.
5. **Карта памяти** (Memory Mapping) — для динамических библиотек и файлов.

**28. Linux: malloc и calloc**

Память, выделенная с помощью **malloc** и **calloc**, размещается в области **heap** адресного пространства.