|  |
| --- |
| **using System; // Подключение пространства имен для базовых классов**  **using System.Collections.Generic; // Подключение пространства имен для работы с обобщенными коллекциями**  **using System.Threading; // Подключение пространства имен для работы с потоками**  **class Program**  **{**  **static void Main(string[] args)**  **{**  **// Создаем список для хранения объектов типа Big**  **List<Big> bigObjects = new List<Big>();**  **// Бесконечный цикл**  **while (true)**  **{**  **// Создаем новый объект Big и добавляем его в список**  **bigObjects.Add(new Big());**  **// Получаем общее количество используемой памяти в байтах и конвертируем в мегабайты**  **long memoryUsed = GC.GetTotalMemory(false) / (1024 \* 1024);**  **// Выводим текущее использование памяти**  **Console.WriteLine($"Используемая память: {memoryUsed} MB");**  **// Задержка выполнения на 5 секунд**  **Thread.Sleep(5000);**  **}**  **}**  **}**  **// Определение класса Big**  **class Big**  **{**  **// Поле для хранения массива байтов**  **public byte[] Data;**  **// Конструктор класса Big**  **public Big()**  **{**  **// Выделяем 128 мегабайт памяти для массива**  **Data = new byte[128 \* 1024 \* 1024];**  **}**  **}** |

**6. Программа не будет работать бесконечно в реальных условиях, так как она в конечном итоге столкнется с ограничениями памяти.**

**7. рабочий набор увеличивается** и становится практически равным выделенной памяти. Это связано с активным использованием памяти, что заставляет операционную систему хранить данные в оперативной памяти.

8.  Частный рабочий набор увеличивается линейно, так как каждый экземпляр выделяет отдельные массивы для объектов.

 Общий рабочий набор растет медленнее, так как код программы и системные библиотеки разделяются между экземплярами.

* + **cat /proc/$(pidof lab)/maps**(Показывает адресное пространство процесса.
  + Секции lab:
  + .text (код): r-xp
  + .data (инициализированные данные): rw-p
  + .bss (неинициализированные данные): rw-p
  + Есть также стек, куча и библиотеки (libc.so.6, ld-linux-x86-64.so.2).
  + **pmap $(pidof lab)** Упрощённая структура:
  + Адреса секций совпадают с /proc/[pid]/maps.
  + Общий размер: 2460 KB.
  + **objdump -f lab** предоставил начальный адрес исполнения (0x1070) и информацию о секциях ELF-файла.

Если бы вы хотели получить значение по адресу 0x260030A5, оно будет соответствовать 5-му байту слова 03B7F000, которое в шестнадцатеричном представлении:

* 03B7F000:
  + **0-й байт:** 00
  + **1-й байт:** 00
  + **2-й байт:** B7
  + **3-й байт:** 03
  + **4-й байт:** F0

Задание 05

Ответьте на следующие вопросы

14. Поясните понятие «виртуальная память».

Виртуальная память - это концепция, позволяющая программам исполняться, как если бы у каждой из них было свое собственное пространство оперативной памяти.

15. Поясните понятие «свопинг».

Свопинг (англ. swapping) - это процесс передачи данных, хранящихся в оперативной памяти, на диск и обратно. Когда операционная система замечает, что физическая память заполняется, она может переместить некоторые неиспользуемые данные на диск в файл подкачки, освобождая физическую память для новых данных.

16. Поясните понятие «страничная память».

Страничная память - это механизм управления памятью, используемый операционными системами. В рамках концепции страничной памяти виртуальное адресное пространство программы разбивается на небольшие блоки, называемые страницами.

17. Поясните понятие MMU.

MMU (Memory Management Unit):

MMU - это устройство, обычно встроенное в процессор, которое отвечает за преобразование виртуальных адресов, используемых программами, в соответствующие физические адреса в оперативной памяти.

18. Поясните понятие TLB.

TLB (Translation Lookaside Buffer) - это кэш ассоциативной памяти, используемый MMU для ускорения преобразования виртуальных адресов в физические адреса.

19. Какая информация содержится в строке таблицы страниц

Строка таблицы страниц содержит информацию о соответствии виртуальных страниц программы физическим фреймам памяти.

20. Поясните принцип применения хэш-таблиц.

Хэш-таблицы используются для эффективного хранения и поиска данных. Они основаны на хэш-функциях, которые преобразуют ключи данных в хэш-коды. Хэш-таблица состоит из массива ячеек, где каждая ячейка содержит список элементов с одинаковыми хэш-кодами.

21. Поясните применение «инвертированной таблицы физических» страничной памяти.

Инвертированная таблица физических страниц - это особый способ организации страничной памяти, где вместо того, чтобы хранить соответствия виртуальных страниц программы и физических фреймов в каждом процессе, используется общая таблица, известная как инвертированная таблица.

22. Windows: перечислите области адресного пространства (от младших к старшим адресам) и поясните их назначения.

а) Нижняя область адресного пространства (Lower Address Space) - эта область подразделяется на несколько сегментов:

- Сегмент кода (Code Segment) - в этом сегменте размещается исполняемый код программы.

- Сегмент данных (Data Segment) - в этом сегменте размещаются глобальные и статические данные программы.

- Сегмент стека (Stack Segment) - в этом сегменте размещается стек вызовов функций и локальные переменные.

б) Область адресного пространства, зарезервированная для системы (Reserved System Space) - эта область зарезервирована для нужд операционной системы и не доступна для пользовательских приложений.

в) Область адресного пространства, зарезервированная для пользовательских приложений (Reserved User Space) - это область адресного пространства, которая доступна для пользовательских приложений для выделения памяти и размещения данных.

23. Windows: какой стандартный начальный размер области heap?

Стандартный начальный размер области heap в Windows зависит от версии операционной системы и настроек системы. В общем случае, стандартный начальный размер heap составляет 1 мегабайт (1 MB).

24. Windows: какой стандартный размер области памяти stack?

Стандартный размер области памяти stack в Windows зависит от версии операционной системы и настроек системы. В общем случае, стандартный размер stack составляет 1 мегабайт (1 MB).

25. Windows: что такое «рабочее множество»? поясните принцип управления рабочим множеством с помощью OS API.

В контексте Windows термин "рабочее множество" (working set) относится к набору физических страниц памяти, используемых процессом в данный момент. Управление рабочим множеством осуществляется с помощью OS API, таких как функции SetProcessWorkingSetSize и QueryWorkingSet из Windows API. С помощью SetProcessWorkingSetSize можно установить ограничения на размер рабочего множества процесса, а с помощью QueryWorkingSet можно получить информацию о текущем состоянии рабочего множества.

26. Windows: что такое «heap»? Что такое «heap процесса»? Что такое «пользовательская heap»? Поясните принцип устройства heap.

Heap представляет собой блоки памяти различного размера, которые могут быть выделены и освобождены во время выполнения программы.

Heap процесса (process heap) - это основной heap, создаваемый для каждого процесса при его запуске.

Пользовательская heap (user heap) - это heap, создаваемый пользовательским приложением для выделения и управления памятью.

Устройство heap основано на списке блоков памяти различных размеров, известном как список свободных блоков (free list). При запросе памяти heap ищет подходящий блок в списке свободных блоков и выделяет его для приложения. При освобождении памяти блок возвращается в список свободных блоков для последующего использования.

27. Linux: перечислите области адресного пространства (от младших к старшим адресам) и поясните их назначения.

а) Нижняя область адресного пространства:

- Область кода (Code Segment) - в этой области размещается исполняемый код программы.

- Область данных (Data Segment) - в этой области размещаются глобальные и статические данные программы.

- Область BSS (Block Started by Symbol) - в этой области размещаются неинициализированные глобальные и статические переменные.

- Область стека (Stack Segment) - в этой области размещается стек вызовов функций и локальные переменные.

б) Область адресного пространства, зарезервированная для библиотек и разделяемых объектов (Shared Libraries and Shared Objects) - в этой области размещаются библиотеки, используемые программой, и разделяемые объекты, которые могут быть загружены и используемы несколькими процессами одновременно.

в) Область адресного пространства, зарезервированная для отображения файлов в память (Memory-Mapped Files) - в этой области размещаются файлы, которые могут быть отображены в память и использованы как обычная память. Это позволяет обращаться к содержимому файла напрямую через указатели и использовать его без необходимости чтения и записи на диск.

г) Область адресного пространства, зарезервированная для стека сигналов (Signal Stack) - в этой области размещается стек, используемый для обработки сигналов, которые могут быть отправлены процессу.

д) Область адресного пространства, зарезервированная для динамической загрузки библиотек (Dynamic Linking) - в этой области размещаются библиотеки, которые загружаются динамически во время выполнения программы.

е) Область адресного пространства, зарезервированная для кучи (Heap Segment) - в этой области происходит выделение и управление динамической памятью с помощью функций, таких как malloc и calloc.

28. Linux: в какой части адресного пространства выделяется памяти с помощью функций malloc, calloc?

В Linux память, выделяемая с помощью функций malloc и calloc, выделяется в области адресного пространства, называемой "кучей" (Heap Segment). Куча представляет собой область памяти, используемую для динамического выделения и освобождения памяти во время выполнения программы.

Область адресного пространства, выделенная для кучи, находится в верхней части адресного пространства процесса и растет в направлении старших адресов при выделении памяти.