Разпределяне на паметта без използване на дисково пространство

1. Въведение

Всеки компютър използва памет, за да работят програмите. Тази памет е оперативната памет (RAM). Тя е бърза, но ограничена. При някои системи и програми се изисква да разпределяме паметта много ефективно, без да използваме дискове за съхранение.

Днес ще научим как това става, какви методи има и какво е важно да знаем, за да не "изчерпим" паметта.

---

2. Какво е разпределяне на паметта?

Представете си, че паметта на компютъра е като една голяма тетрадка. Когато пишем в нея, трябва да знаем:

1. **Къде да пишем** – т.е. коя част от паметта ще ползваме.
2. **Колко място ни трябва** – за да не заемаме повече, отколкото ни е нужно.
3. **Кога да изтрием написаното** – за да може други програми да ползват освободената част.

---

3. Видове разпределяне на паметта

3.1. Статично разпределяне на паметта

- **Какво е това?**
  При този метод решаваме колко памет ще ползваме още преди програмата да започне да работи.
- **Пример:**
  Ако знаете, че ще ползвате 100 реда от тетрадката, ще ги запазите предварително.
- **Пример в програмирането:**

```
int numbers[100]; // Резервира 100 числа в паметта
```

- **Предимства:**
  - Много бързо и лесно.
  - Паметта се знае предварително.
- **Недостатъци:**
  - Ако запазите твърде много памет, тя може да остане неизползвана.
  - Ако ви трябва повече памет, няма как да добавите допълнително.

3.2. Динамично разпределяне на паметта

- **Какво е това?**
  Тук паметта се разпределя, докато програмата работи. Ако ви трябва още място, го "взимате", когато ви е необходимо.
- **Пример:**
  Представете си, че вместо да запазите 100 реда, ще започнете с 10. Ако те се запълнят, ще добавите още.
- **Пример в програмирането: ИЗПОЛЗАВНЕ НА СПИСЪЦИ**
- **Предимства:**

  - Гъвкавост – използвате само толкова памет, колкото ви трябва.
  - Не запазвате излишно място.

- **Недостатъци:**

  - Ако забравите да "освободите" паметта, тя остава блокирана (т.нар. изтичане на памет).
  - По-бавна е от статичното разпределяне.

3.3 Стек и купчина (Stack and Heap)

4. **Стек:** Стекът е универсален инструмент, който решава специфични задачи в програмирането и в реалния свят. Той е полезен, защото помага да се запомнят действия или данни, които трябва да бъдат обработени в обратен ред.

А) Запазва историята на действията

- Стекът съхранява последователност от действия и позволява да се връщаме назад.
  Пример:

  - В уеб браузъра, когато натиснем бутона "Назад", той използва стек, за да върне предишната страница.
  - Последната посетена страница е първата, която се маха (pop).

Б) Управлява задачи в правилен ред

- Ако имаме задачи, които трябва да изпълним в реда, в който са въведени (но в обратен ред на изпълнение), стекът е идеалното решение.
  Пример:

  - Представете си списък от задачи: първо трябва да добавите нещо в програмата, после да го проверите, и накрая да го тествате. Ако използвате стек, ще започнете от последната задача – "тест", и ще завършите с първата.

В) Обработва данни временно

- В програмирането стекът често се използва като временно място за съхранение на данни, докато се изпълнява някакъв алгоритъм.
  Пример:

    - Когато използвате калкулатор за сложни изрази, стекът временно съхранява числата и операциите, докато изразът се изчислява.

Г) Поддържа рекурсия

- В програмирането рекурсивните функции (функции, които извикват сами себе си) разчитат на стек, за да запазват състоянието си.
  Пример:

    - Когато компютърът изчислява факториел (например 5! = 5 × 4 × 3 × 2 × 1), той използва стек, за да запомни какво трябва да умножи, докато стигне до дъното на задачата.

Д) Помага при "отмяна на действия"

- Стекът е идеален за действия като "Undo" (отмяна на последното действие).
  Пример:

    - В текстов редактор като Word, когато отмените последните промени, те се премахват от стек, където са били съхранение

5. Как да използваме паметта правилно?

А) Планирайте нуждите си:

- Преценете дали ви е нужна статична или динамична памет.

Б) Освобождавайте паметта:

- Ако сте заделили динамична памет, не забравяйте да я освободите

В) Избягвайте грешки:

- Никога не се опитвайте да пишете в "освободена" памет. Това може да доведе до срив на програмата.

Заключение

Разпределянето на паметта е много важно за създаването на ефективни програми. Изборът на метод зависи от задачите, които трябва да решите. Ако се научите да разпределяте памет правилно, ще създавате по-стабилни и бързи програми.
