

Содержание

- сегментация памяти** 3
- stack*** 4
- heap*** 5
- const + global*** 5
- code segment*** 5

сегментация памяти

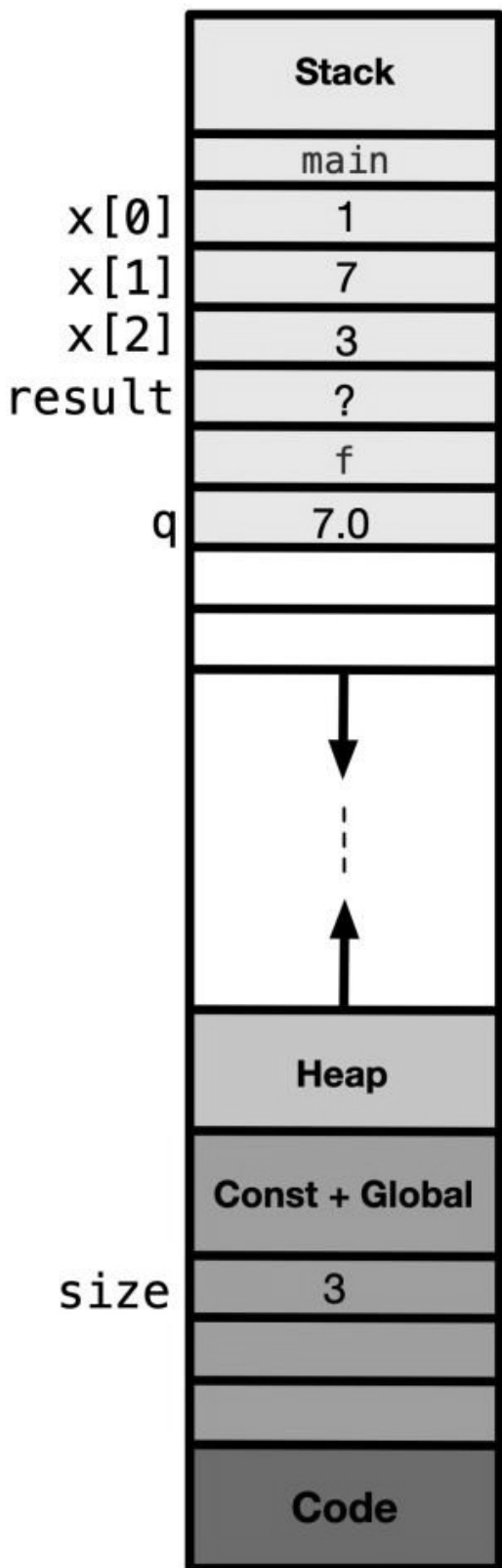
Рассмотрим пример:

```
#include <stdio.h>
double f(int[]);

int size = 3;

int main(void) {
    int x[3] = {1, 7, 3};
    double result = f(x);
    return 0;
}

double f(int list[]) {
    double q = 7.0;
    return q;
}
```



```
1 #include <stdio.h>
2 double f(int[]);
3
4 int size = 3;
5
6 int main(void) {
7     int x[3] = {1, 7, 3};
8     double result = f(x);
9     return 0;
10 }
11 double f(int list[]) {
12     double q = 7.0;
13     return q;
14 }
```

stack

Сегмент стека располагается в верхней части структуры памяти и растет вниз при вызовах функций и создании локальных переменных. Этот сегмент содержит все локальные переменные, параметры функций и адреса возврата. При каждом вызове функции создается новый фрейм стека, который хранит ее локальные данные.

В данном примере сегмент стека хранит локальный массив `x`, содержащий значения `{1, 7, 3}`, а также такие переменные, как `result`, `f` и `q`, которые существуют в области видимости соответствующих функций.

Стек работает по принципу `last-in-first-out` (последним пришел — первым ушел), автоматически управляя выделением и освобождением памяти при вызовах и возврате функций. Когда выполняется функция `main`, она создает пространство для массива `x` и переменной `result`, в то время как функция `f` создает собственный фрейм стека, содержащий локальную переменную `q`, инициализированную значением `7.0`.

Такое автоматическое управление памятью делает стек эффективным для временного хранения данных, но ограничивает время жизни переменных областью их видимости.

heap

Сегмент динамической памяти (`heap`) предоставляет возможности динамического распределения памяти, хотя в этом простом примере он не используется. Расположенный между стеком и нижними сегментами памяти, `heap` растет вверх и позволяет программам запрашивать память во время выполнения с помощью функций `malloc()` и `free()`.

const + global

Сегмент констант и глобальных переменных хранит переменные, которые существуют на протяжении всего выполнения программы, например, глобальную переменную `size` со значением `3`. Этот сегмент обычно содержит инициализированные глобальные переменные, статические переменные и строковые литералы, которые должны оставаться доступными из нескольких функций или на протяжении всего времени существования программы.

code segment

Сегмент кода располагается в нижней части структуры памяти и содержит скомпилированные машинные инструкции для всех функций программы. Этот сегмент обычно доступен только для чтения, чтобы предотвратить случайное изменение программного кода во время выполнения.

Функция `f` демонстрирует простую реализацию, которая всегда возвращает значение `7.0` с помощью локальной переменной `q`, в то время как функция `main` организует поток программы, инициализируя массив, вызывая функцию `f` и сохраняя возвращенное значение.

Организация памяти обеспечивает хранение каждого типа данных в соответствующем сегменте на основе его области видимости, времени жизни и шаблонов использования, обеспечивая эффективность и безопасность выполнения программы.

From:

<https://wiki.radi0.cc/> - radi0wiki

Permanent link:

https://wiki.radi0.cc/c:c_ultimate_guide:mem_segmentation

Last update: **2025/11/09 12:07**

