# Экстремальный Си

## Глава 4. Структура памяти процесса

В этой главе поговорим о памяти и её структуре внутри процесса. Управления памятью – крайне важная тема для любого программиста на языке Си.

В Си управление памятью полностью ручное. Более того, вся ответственность за выделение областей памяти и их освобождение после того, как они больше не нужны, лежит на программисте.

В этой главе рассмотрены следующие вопросы:

* структура памяти типичного процесса;
* статическая и динамическая схема размещения в памяти;
* сегменты, встречающиеся в статической и динамической схемах;
* инструменты и команды, помогающие обнаружить сегменты и просматривать их содержимое – как в объектном файле, так и глубоко внутри активного процесса

### Внутреннее устройство памяти процесса

*Процесс* – активная запущенная программа, которая загружена в память и имеет уникальный *идентификатор* (process identifier, PID). ОС полностью контролирует создание и загрузку новых процессов.

Процесс перестаёт быть активным либо в результате нормального завершения, либо при получении сигнала типа **SIGTERM**, **SIGINT** или **SIGKILL**, который в итоге заставляет его прекратить работу. Сигналы **SIGTERM** и **SIGINT** можно игнорировать, но **SIGKILL** останавливает процесс немедленно и принудительно.

Память типичного процесса делится на несколько частей, которые называются *сегментами*. Каждый из них представляет собой область памяти с определённой задачей, предназначенную для хранения данных конкретного типа:

* сегмент неинициализированных данных или BSS (block started by symbol – блок, начинающийся с символа);
* сегмент данных;
* текстовый сегмент или сегмент кода;
* сегмент стека;
* сегмент кучи.

### Исследование структуры памяти

Одни из сегментов памяти находятся в исполняемом файле, а другие создаются динамически на этапе выполнения, при создании процесса.

*Процесс* – динамическая сущность, выполняемая внутри операционной системы, в то время как исполняемый объектный файл – просто набор данных с подготовленной начальной структурой, и на её основе создаётся будущий процесс. Действительно, некоторые сегменты в структуре памяти активного процесса берутся непосредственно из исполняемого файла, а остальные создаются динамически во время его загрузки. Первые называют *статической схемой размещения в памяти*, а вторые – *динамической*.

Содержимое статической схемы заранее записывается в исполняемый объектный файл во время компиляции. Поэтому содержимое статической памяти можно предсказать по исходному коду или скомпилированному объектному файлу.

Динамическая схема формируется инструкциями процесса, которые выделяют память для переменных и массивов и изменяют её в соответствии с логикой программы. Динамическую схему можно исследовать только запустив программу. Кроме того, при каждом выполнении одной и той же программы содержимое динамической памяти может меняться. Т.е. исследовать динамическое содержимое процесса можно только пока он работает.

### Исследование статической схемы размещения в памяти

Команда **size** – с её помощью можно вывести статическую схему размещения в памяти исполняемого объектного файла.

### Сегмент BSS

BSS (*block started by symbol*) – с давних пор так обозначают области памяти, зарезервированные для неинициализированных машинных слов. Сегмент BSS фактически предназначен для хранения либо неинициализированных, либо обнулённых глобальных переменных.

### Сегмент Data

Чтобы продемонстрировать какого рода содержимое содержится в сегменте Data, объявим больше глобальных переменных, однако на сей раз инициализируем их с помощью ненулевых значений.

### Сегмент Text

Поскольку все машинные инструкции программы находятся в сегменте Text (или Code), он должен находиться в исполняемом объектном файле — а именно, в его статической схеме размещения. Процессор извлекает эти инструкции и выполняет их во время работы процесса.

### Исследование динамической схемы размещения в памяти

Динамическая схема размещения находится в памяти процесса и существует до тех пор, пока он не завершится. Процедурой запуска исполняемого объектного файла занимается программа под названием *«загрузчик»*.

Если коротко, то в памяти активного процесса должно быть пять сегментов, три из которых копируются непосредственно из статической схемы размещения исполняемого объектного файла, а остальные два создаются с нуля и называются стеком и кучей. Последние являются динамическими сегментами и существуют только в период работы процесса. Это значит, вы не найдете никакого упоминания о них в исполняемом объектном файле.

*Стек* — область памяти, в которой по умолчанию выделяется место для переменных. Она имеет ограниченный размер, поэтому большие объекты в ней хранить нельзя. Для сравнения, *куча* — более крупная и гибкая область памяти, в которой могут поместиться большие объекты и массивы. Для работы с *кучей* нужен отдельный API, с которым мы познакомимся чуть позже.

**Динамическая схема размещения — не то же самое, что *динамическое выделение памяти*. Не следует путать эти два понятия, поскольку они имеют разный смысл!**

### Отражение памяти

Напишем код, который выполняется бесконечно долго. Таким образом получим процесс, который позволит нам изучить его структуру памяти.