**Задание №3.4 в рамках вычислительного практикума.**

**Представление в памяти структур и объединений**

**Локальные переменные**

Исходный код:

#include <stdio.h>

int main(void)

{

    int a = -20;

    double b = 3.14;

    char c = 'a';

    short d = 2;

    return 0;

}

Дамп всех локальных переменных:

// int a

(gdb) x/4xb &a

0x7fffffffdce4: 0xec 0xff 0xff 0xff

// double b

(gdb) x/8xb &b

0x7fffffffdce8: 0x1f 0x85 0xeb 0x51 0xb8 0x1e 0x09 0x40

// char c

(gdb) x/1xb &c

0x7fffffffdce1: 0x61

// short d

(gdb) x/2xb &d

0x7fffffffdce2: 0x02 0x00

Дамп памяти, который содержит все локальные переменные:

(gdb) x/15xb &c

0x7fffffffdce1: 0x61 // c 0x02 0x00 // d 0xec 0xff 0xff 0xff 0x1f // a

0x7fffffffdce9: 0x85 0xeb 0x51 0xb8 0x1e 0x09 0x40 // b

Сведения о переменных:

|  |  |  |
| --- | --- | --- |
| Имя | Размер | Адрес |
| a | 4 | 0x7fffffffdce4 |
| b | 8 | 0x7fffffffdce8 |
| c | 1 | 0x7fffffffdce1 |
| d | 2 | 0x7fffffffdce2 |

Вывод: компилятор поменял расположение переменных в памяти для минимизации паддинга и оптимизации выравнивания, если бы переменные располагались в порядке инициализации, то потребовалось бы дополнительное выравнивание и паддинг. Компилятор расположил переменные в порядке: char c → short d → int a → double b.

**Структуры**

Описание структуры и структурной переменной:

#include <stdbool.h>

#include <stdio.h>

#define MAX\_WORD\_LEN 25

typedef struct

{

    char bookName[MAX\_WORD\_LEN];

    char bookAuthor[MAX\_WORD\_LEN];

    int bookPages;

    bool isReaded;

} Book;

int main(void)

{

    Book book\_about\_c = {

.bookName = "The C Programmin Language",

.bookAuthor = "K&R",

.bookPages = 274,

.isReaded = false};

    return 0;

}

Дамп памяти структурной переменной:

(gdb) p sizeof(book\_about\_c)

$1 = 60

(gdb) x/60xb &book\_about\_c

0x7fffffffdca0: 0x54 0x68 0x65 0x20 0x43 0x20 0x50 0x72

0x7fffffffdca8: 0x6f 0x67 0x72 0x61 0x6d 0x6d 0x69 0x6e

0x7fffffffdcb0: 0x20 0x4c 0x61 0x6e 0x67 0x75 0x61 0x67

0x7fffffffdcb8: 0x65 0x00 0x4b 0x26 0x52 0x00 0x00 0x00

0x7fffffffdcc0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x7fffffffdcc8: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x7fffffffdcd0: 0x00 0x00 0x00 0x00 0x12 0x01 0x00 0x00

0x7fffffffdcd8: 0x00 0x5a 0xfe 0xf7

Сведения о полях структурной переменной

|  |  |  |
| --- | --- | --- |
| Имя | Размер | Адресс |
| bookName | 26 | 0x7fffffffdca0 |
| bookAuthor | 26 | 0x7fffffffdcba |
| bookPages | 4 | 0x7fffffffdcd4 |
| isReaded | 4 (выравнивание) | 0x7fffffffdcd8 |

Адрес поля структуры не определяется напрямую его размером, а зависит от выравнивания и порядка объявления.

Поле, влияющее на выравнивание структуры: поле с наибольшим требованием к выравниванию. Адрес структуры должен быть кратен её выравниванию, что гарантирует корректное размещение всех полей.

Описание упакованной структуры:

#pragma pack(push, 1)

typedef struct

{

    char bookName[MAX\_WORD\_LEN];

    char bookAuthor[MAX\_WORD\_LEN];

    int bookPages;

    bool isReaded;

} Book;

#pragma pack(pop)

// все остальное без изменений

Дамп памяти упакованной структурной перменной:

(gdb) x/57xb &book\_about\_c

0x7fffffffdca0: 0x54 0x68 0x65 0x20 0x43 0x20 0x50 0x72

0x7fffffffdca8: 0x6f 0x67 0x72 0x61 0x6d 0x6d 0x69 0x6e

0x7fffffffdcb0: 0x20 0x4c 0x61 0x6e 0x67 0x75 0x61 0x67

0x7fffffffdcb8: 0x65 0x00 0x4b 0x26 0x52 0x00 0x00 0x00

0x7fffffffdcc0: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x7fffffffdcc8: 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

0x7fffffffdcd0: 0x00 0x00 0x00 0x00 0x12 0x01 0x00 0x00

0x7fffffffdcd8: 0x00

Сведения о полях упакованной структурной переменной:

|  |  |  |
| --- | --- | --- |
| Имя | Размер | Адресс |
| bookName | 26 | 0x7fffffffdca0 |
| bookAuthor | 26 | 0x7fffffffdcba |
| bookPages | 4 | 0x7fffffffdcd4 |
| isReaded | 1 | 0x7fffffffdcd8 |

В моем случае от перестановки полей bookPages и isReaded размер структуры не изменился.

Убрались 3 байта у поля isReaded(раньше выравнивалось до 4 из-за int), из-за того что мы упаковали структуру.

**Объеденения**

Описание объединения и инициализации одного из его полей:

#include <stdio.h>

typedef union {

    int age;

    char gender;

    double height;

} Human;

int main(void)

{

    Human you = {you.gender = 'm'};

    return 0;

}

Представление объединения в памяти:

(gdb) p sizeof(&you)

$1 = 8

(gdb) x/8xb &you

0x7fffffffdce8: 0x6d 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Дамп памяти объединения (поле age имеет значение 25)

(gdb) x/8xb &you

0x7fffffffdce8: 0x19 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Дамп память объединения (поле height имеет значение 183.761)

(gdb) x/8xb &you

0x7fffffffdce8: 0xb7 0x00 0x00 0x00 0x5a 0xf8 0x66 0x40