Смирнов Иван ИУ7-22Б - 2023г.

Отчет

Задание №3.4

Отладка

Целью работы является изучение расположения в памяти локальных переменных и представления структур.

1. Локальные переменные

1) Код программы, в котором описано несколько локальных переменных разных типов:

```
#include <stdio.h>
#define OK 0
void func()
{
	int left = 10;
	short right = 100;
	double middle = 0;
	printf("%d\n", right-left);
}

int main(void)
{
	func();
	return OK;
}
```

В функции func() объявлены 3 переменные разных типов.

2) Дамп памяти данных локальных переменных (в gdb):

```
Breakpoint 1, func () at main.c:10
10		printf("%d\n", right-left);

(gdb) print (&left)
$3 = (int *) 0x7fffffffdf14
(gdb) print (&right)
$4 = (short *) 0x7fffffffdf12
(gdb) print (&middle)
$5 = (double *) 0x7fffffffdf18

(gdb) print sizeof(left)
$3 = 4
(gdb) x/4xb &left
0x7fffffffdf1c:	0x0a	0x00	0x00	0x00
(gdb) print sizeof(right)
$4 = 2
(gdb) x/2xb &right
0x7fffffffdf1a:	0x64	0x00
```

```
(gdb) print sizeof(middle)
$1 = 8
(gdb) print (&middle)
$2 = (double *) 0x7fffffffdf18
(gdb) x/8xb &middle
0x7fffffffdf18:    0x00    0x00    0x00    0x00    0x00    0x00    0x00    0x00
```

3) Таблица:

Имя переменной	**Размер (в Б)**	**Значение адреса**
Left	4	0x7fffffffdf14
Right	2	0x7fffffffdf12
Middle	8	0x7fffffffdf18

4) Значение адреса переменной должно быть кратно его размерности (размерность см. в табл.). В данном случае переменные в памяти располагаются “друг за другом”, занимая своё количество байт.

2. Структурный тип данных

1) Структура, содержащая несколько полей разных типов:

```
#include <stdio.h>
#define OK 0

int main(void)
{
        struct test
        {
                char a;
                int b;
                double c;
        } t;
        return OK;
}
```

2) Дамп памяти:

```
Breakpoint 1, main () at main.c:13
13              return OK;
(gdb) print sizeof(t)
$1 = 16
(gdb) print &t
$2 = (struct test *) 0x7fffffffdf20
(gdb) print &(t.a)
$7 = 0x7fffffffdf20 ""
(gdb) print &(t.b)
$8 = (int *) 0x7fffffffdf24
(gdb) print &(t.c)
$9 = (double *) 0x7fffffffdf28
(gdb) print sizeof(t.c)
$10 = 8
(gdb) print sizeof(t.b)
$11 = 4
```

```
(gdb) print sizeof(t.a)
$12 = 1
(gdb) x/16xb &t
0x7fffffffdf20:  0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdf28:  0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
(gdb) x/1xb &t.a
0x7fffffffdf20:  0x00
(gdb) x/4xb &t.b
0x7fffffffdf24:  0x00  0x00  0x00  0x00
(gdb) x/8xb &t.c
0x7fffffffdf28:  0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
```

3) Таблица:

Имя поля	**Размер (в Б)**	**Значение адреса**
A	1	0x7fffffffdf20
B	4	0x7fffffffdf24
C	8	0x7fffffffdf28

Значение адреса поля должно быть кратно его размерности (размерность см. в табл.).

4) Переменная структурного типа располагается по адресу 0x7fffffffdf20 и занимает 16 байт. Переменная структурного типа располагается по данному адресу из-за поля самой большей размерности (в данном случае максимальный размер имеет тип double - 8). Выравнивание произошло так же из-за данного поля (учитывая выравнивание поля a и b заняли первые 8 байт, а поле c - вторые 8 байт).
5) Выполним первые 4 пункта с *упакованной* структурой
   5.1) Программа:

```
#include <stdio.h>
#define OK 0
int main(void)
{
    #pragma pack(push, 1)
    struct test
    {
            double c;
            int b;
            char a;
    } t;
    #pragma pack(pop)
    return OK;
}
```

   5.2) Дамп памяти:

```
Breakpoint 1, main () at main.c:15
15              return OK;
(gdb) print sizeof(t)
```

```
$1 = 13
(gdb) print &t
$2 = (struct test *) 0x7fffffffdf20
(gdb) print &t.c
$3 = (double *) 0x7fffffffdf20
(gdb) print &t.b
$4 = (int *) 0x7fffffffdf28
(gdb) print &t.a
$5 = 0x7fffffffdf2c ""
(gdb) print &t+1
$6 = (struct test *) 0x7fffffffdf2d
(gdb) x/13xb &t
0x7fffffffdf20:  0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
0x7fffffffdf28:  0x00  0x00  0x00  0x00  0x00
(gdb) x/8xb &t.c
0x7fffffffdf20:  0x00  0x00  0x00  0x00  0x00  0x00  0x00  0x00
(gdb) x/4xb &t.b
0x7fffffffdf28:  0x00  0x00  0x00  0x00
(gdb) x/1xb &t.a
0x7fffffffdf2c:  0x00
```

5.3) Таблица:

Имя поля	**Размер (в Б)**	**Значение адреса**
A	1	0x7fffffffdf2c
B	4	0x7fffffffdf28
C	8	0x7fffffffdf20

5.4) Переменная структурного типа располагается по адресу 0x7fffffffdf2c и занимает 13 байт. Переменная структурного типа располагается по данному адресу из-за поля самой большей размерности (в данном случае максимальный размер имеет тип double - 8). Так как структура упакована, то выравнивания не произошло, соответственно дополнительные байты не были заняты, а значит структура весит double(8)+int(4)+char(1)=13 байт.

6) Чтобы занимаемых выравниванием битов было минимально, можно переставить поля структуры в отсортированном (в прямом или обратном порядке) по размеру полей виде. В данном случае можно объявить поля в порядке: a, b, c или c, b, a.

```
Breakpoint 1, main () at main.c:13
13                  return OK;
(gdb) print t
$1 = { a = 0 '\000', b = 0, c = 0}
Breakpoint 1, main () at main.c:13
13                  return OK;
(gdb) print t
$1 = {c = 0, b = 0, a = 0 '\000'}
```

7) В данной структуре 'завершающее' выравнивание равно 3 байтам, так как структура состоит полей типов double(8), int(4), char(1), то есть в сумме поля занимают 13 байт, но без упаковки структуры

будет сделано завершающее выравнивание до следующего слова и соответственно будет заполнено ещё 3 байта - то есть в сумме вся структурная переменная будет занимать 16 байт.