Паламарчук Андрей ИУ7-23Б

Выравнивание переменных

1. Описание нескольких локальных переменных разных типов.

Программа:

```
#include <stdio.h>

int main(void)
{
    short sh_var = 1;
    int int_var = 2;
    long long_var = 3;
    long long ll_var = 4;
    float float_var = 5;
    double double_var = 6;

    return 0;
}
```

2. Дамп памяти.

```
$ gcc -std=c99 -g3 main.c
$ gdb ./a.out
GNU gdb (GDB) 13.1

...
Reading symbols from ./a.out...
(gdb) break 13
Breakpoint 1 at 0x1154: file main.c, line 13.
(gdb) run
Starting program: /home/andrey/practic_PTP/Task_3.4/a.out

Breakpoint 1, main () at main.c:13
13          return 0;


(gdb) print &sh_var
$1 = (short *) 0x7fffffffd92e
(gdb) print &int_var
$2 = (int *) 0x7fffffffd930
(gdb) print &long_var
$3 = (long *) 0x7fffffffd938
(gdb) print &ll_var
$4 = (long long *) 0x7fffffffd940
(gdb) print &float_var
$5 = (float *) 0x7fffffffd934
(gdb) print &double_var
$6 = (double *) 0x7fffffffd948


(gdb) x /34tb 0x7fffffffd92e
0x7fffffffd92e: 00000001        00000000        00000010
00000000        00000000        00000000        00000000
00000000
```

```
0x7fffffffd936: 10100000        01000000        00000011
00000000        00000000        00000000        00000000
00000000
0x7fffffffd93e: 00000000        00000000        00000100
00000000        00000000        00000000        00000000
00000000
0x7fffffffd946: 00000000        00000000        00000000
00000000        00000000        00000000        00000000
00000000
0x7fffffffd94e: 00011000        01000000
(gdb)
```

3. Таблица сравнения

Проверим размеры для нашей машины

```
#include <stdio.h>

int main(void)
{
    printf("%d\n", sizeof(short));
    printf("%d\n", sizeof(int));
    printf("%d\n", sizeof(long));
    printf("%d\n", sizeof(long long));
    printf("%d\n", sizeof(float));
    printf("%d\n", sizeof(double));

    return 0;
}
```

Вывод:

```
2
4
8
8
4
8
```

Имя переменной	Размер	Значение адреса
sh_var	2	0x7fffffffd92e
int_var	4	0x7fffffffd930
long_var	8	0x7fffffffd938
ll_var	8	0x7fffffffd940
float_var	4	0x7fffffffd934
double_var	8	0x7fffffffd948

Из этого мы можем сделать вывод, что чем меньше размер переменной, тем меньше её адрес.

Изучение представления структуры в памяти

1. Описание структуры, содержащей несколько полей разного типа.

Программа:

```
#include <stdio.h>

struct s_
{
    char ch_var;
    int int_var;
    double double_var;
};

int main(void)
{
    struct s_ a = {1, 2, 3.0};

    return 0;
}
```

2. Дамп памяти, который содержит эту структуру

```
$ gcc -std=c99 -g3 main_.c
$ gdb ./a.out
GNU gdb (GDB) 13.1

...
Reading symbols from ./a.out...
(gdb) break 15
Breakpoint 1 at 0x1172: file main_.c, line 15.
(gdb) run
Starting program: /home/andrey/practic_PTP/Task_3.4/a.out

Breakpoint 1, main () at main_.c:15
15          return 0;


(gdb) print &a.ch_var
$1 = 0x7fffffffd940 "\001"
(gdb) print &a.int_var
$2 = (int *) 0x7fffffffd944
(gdb) print &a.double_var
$3 = (double *) 0x7fffffffd948

(gdb) print &a
$4 = (struct s_ *) 0x7fffffffd940
(gdb) x /16tb 0x7fffffffd940
0x7fffffffd940: 00000001        00000000        00000000
00000000        00000010        00000000        00000000
00000000
0x7fffffffd948: 00000000        00000000        00000000
00000000        00000000        00000000        00001000
01000000
(gdb)
```

3. Таблица сравнения

Узнаем размеры

```
#include <stdio.h>

struct s_
{
    char ch_var;
    int int_var;
    double double_var;
};

int main(void)
{
    struct s_ a = {1, 2, 3.0};
    printf("%d\n", sizeof(struct s_));
    printf("%d\n", sizeof(a.ch_var));
    printf("%d\n", sizeof(a.int_var));
    printf("%d\n", sizeof(a.double_var));

    return 0;
}
```

Вывод:

```
16
1
4
8
```

Размер структуры 16 байт

Имя поля	Размер	Значение адреса
ch_var	1	0x7fffffffd940
int_var	4	0x7fffffffd944
double_var	8	0x7fffffffd948

Из этого мы можем сделать вывод, что чем меньше размер поля, тем меньше его адрес.

4. Адрес переменной структурного типа и его значение.

```
(gdb) print &a
$4 = (struct s_ *) 0x7fffffffd940
```

0x7fffffffd940 – адрес нашей переменной структурного типа

Он совпадает с переменной ch_var, так как адрес первого поля совпадает с адресом переменной структурного типа.

5. Минимальное место, занимаемое структурой.

Переберем N! факториал комбинаций расположения наших переменных

Таблица соответствия комбинации переменных и размера структуры

Позиция: 1	double	double	int	char	int	char
Позиция: 2	int	char	double	double	char	int
Позиция: 3	char	int	char	int	double	double
Размер	16	16	24	24	16	16

Минимальное место, которое может занимать наша структура без упаковки, равно 16-ти байтам.

6. «Завершающее» выравнивание

У нашей структуры есть «завершающее» выравнивание, которое дополняет нашу char переменную до 4 байт.

«Завершающее» выравнивание:

4 – 1 = **3 байта**

Изучение представления упакованной структуры в памяти

1. Описание структуры, содержащей несколько полей разного типа.

Программа:

```
#include <stdio.h>

struct s_
{
    unsigned char ch_var : 1;
    int int_var;
    double double_var;
};

int main(void)
{
    struct s_ a = {1, 2, 3.0};

    return 0;
}
```

2. Дамп памяти, который содержит эту структуру

```
$ gcc -std=c99 -g3 main_.c
$ gdb ./a.out
GNU gdb (GDB) 13.1

...
Reading symbols from ./a.out...
(gdb) break 14
Breakpoint 1 at 0x113b: file main_.c, line 14.
(gdb) run
Starting program: /home/andrey/practic_PTP/Task_3.4/a.out

Breakpoint 1, main () at main_.c:14
14          return 0;
(gdb) print &a.ch_var
$1 = (unsigned char *) 0x7fffffffd940 "\001"
(gdb) print sizeof(a.ch_var)
$2 = 1
(gdb) print &a.int_var
$3 = (int *) 0x7fffffffd944

(gdb) print sizeof(a.int_var)
$4 = 4
(gdb) print &a.double_var
$5 = (double *) 0x7fffffffd948

(gdb) print sizeof(a.double_var)
$6 = 8

(gdb) print &a
$7 = (struct s_ *) 0x7fffffffd940

(gdb) print sizeof(a)
$8 = 16
(gdb)
```

3. Таблица сравнения

Размер структуры 16 байт

Имя поля	Размер	Значение адреса
ch_var	1	0x7fffffffd940
int_var	4	0x7fffffffd944
double_var	8	0x7fffffffd948

Из этого мы можем сделать вывод, что чем меньше размер поля, тем меньше его адрес.

4. Адрес переменной структурного типа и его значение.

```
(gdb) print &a
$4 = (struct s_ *) 0x7fffffffd940
```

0x7fffffffd940 – адрес нашей переменной структурного типа

Он совпадает с переменной ch_var, так как адрес первого поля совпадает с адресом переменной структурного типа.