Задание №5 в рамках вычислительного практикума.
Представление в памяти строк и массивов строк

Студент: Краснов Леонид
Группа: ИУ7-21Б

Оглавление

Цель работы

Изучить, как в памяти представлена строка языка Си. Показать дамп памяти, который содержит строку полностью. Объясните, какие байты за что отвечают.
Изучите разные способы хранения массива слов в языке Си (Слайд No10 из лекции No5, значения строк должны быть Вашими). Для каждого способа:

Задание

1. Показать дамп памяти, который содержит массив строк полностью (первый способ)

Код программы:

```c
#include <stdio.h>


int main()
{
    char str[][8] = {"Sony", "Samsung", "Apple"};
    return 0;
}
```

Дамп памяти:

```
(gdb) b 7
Breakpoint 1 at 0x778: file main.c, line 7.
(gdb) run
Starting program: /home/x13eav1sx/Рабочий стол/practice5/a.out

Breakpoint 1, main () at main.c:7
7           return 0;
(gdb) p sizeof(str)
$1 = 24
(gdb) x /24xb &str
0xffffffffec18: 0x53    0x6f    0x6e    0x79    0x00    0x00    0x00    0x00
0xffffffffec20: 0x53    0x61    0x6d    0x73    0x75    0x6e    0x67    0x00
0xffffffffec28: 0x41    0x70    0x70    0x6c    0x65    0x00    0x00    0x00
(gdb) p str[0]
$1 = "Sony\000\000\000"
(gdb) p str[1]
$2 = "Samsung"
(gdb) p str[2]
$3 = "Apple\000\000"
```

2. Показать дамп памяти, который содержит массив строк полностью (второй способ)

Код программы

```c
#include <stdio.h>


int main()
{
    char *str[] = {"Sony", "Samsung", "Apple"};
    return 0;
}
```

Дамп памяти

```
(gdb) b 7
Breakpoint 1 at 0x7b8: file main.c, line 7.
(gdb) run
Starting program: /home/x13eav1sx/Рабочий стол/practice5/a.out

Breakpoint 1, main () at main.c:7
7           return 0;
(gdb) p sizeof(arr)
No symbol "arr" in current context.
(gdb) p sizeof(str)
$1 = 24
(gdb) x /24xb str
0xffffffffec18: 0x70    0x08    0xaa    0xaa    0xaa    0xaa    0x00    0x00
0xffffffffec20: 0x78    0x08    0xaa    0xaa    0xaa    0xaa    0x00    0x00
0xffffffffec28: 0x80    0x08    0xaa    0xaa    0xaa    0xaa    0x00    0x00
(gdb) p str[0]
$1 = 0xaaaaaaaa0870 "Sony"
(gdb) p str[1]
$2 = 0xaaaaaaaa0878 "Samsung"
(gdb) p str[2]
$3 = 0xaaaaaaaa0880 "Apple"
```

3. Прокомментировать, что есть что в этих дампах памяти. Почему байты имеют именно такое значение?

В первом случае, строки расположены последовательно, и каждый байт целочисленное число, которое обозначает символ, однако для создания массива строк, каждая строка имеет фиксированную длину, из-за чего недостающие символы заменяются нулем.
Во втором случае, строки имеют не фиксированную длину, из-за чего в каждой строке есть только 1 ноль, который используется как символ конца строки.

4. Рассчитать суммарный размер памяти, который занимает каждая структура данных. Каков размер «полезных» и «вспомогательных» данных?

Код программы

```c
#include "stdio.h"


struct data
{
    int i;
    float f;
    double d;
    long l;
    long long ll;
};


int main(void)
{
    struct data d;
```

```
    d.i = 10;
    d.f = 2.27;
    d.d = 3.14543423;
    d.l = 123231;
    d.ll = 345342423423423;
    return 0;
}
```

Размер структуры

```
(gdb) b 22
Breakpoint 1 at 0x79c: file main.c, line 22.
(gdb) run
Starting program: /home/x13eav1sx/Рабочий стол/practice5/a.out

Breakpoint 1, main () at main.c:22
22          return 0;
(gdb) p sizeof(int)
$1 = 4
(gdb) p sizeof(float)
$2 = 4
(gdb) p sizeof(double)
$3 = 8
(gdb) p sizeof(long)
$4 = 8
(gdb) p sizeof(long long)
$5 = 8
(gdb) p $1+$2+$3+$4+$5
$6 = 32
(gdb) p sizeof(d)
$7 = 32
```

Здесь видно, что размер структуры соответствует сумме размеров типов данных, которые она содержит. Отсюда можно сделать вывод, что полезные данные в структуре занимают все пространство, а вспомогательных данных нет.