Доколин Георгий ИУ7-22Б

Отчет по Проектно-технологической практике (тестированию, отладке и профилированию ПО)
Задание №3.3 в рамках вычислительного практикума.
Представление строк в памяти

Оглавление

Введение

Цель работы: изучение представлениия строки в памяти компьютера при помощи отладчика **gdb**.

Описание строки str и её инициализация

```
#include <stdio.h>

int main(void)
{
   char str[] ="Hello world!";
   printf("%s\n", str);
   return 0;
}
```

Дамп всей строки

Для дампа всей строки воспользуемся командой **x *Nxb address***, где N размер перменной, a address адрес переменной. Размер строки “Hello world!” составляет 13 байт, тк каждый элемент занимает ровно один байт, также можно это проверить при помощи фунции **sizeof.**

```
(gdb) x /13xb str
0x7fffffffdc9b: 0x48    0x65    0x6c    0x6c    0x6f    0x20
0x77    0x6f
0x7fffffffdca3: 0x72    0x6c    0x64    0x21    0x00
```

Пояснения:
Если посмотреть дамп всей строки то можно понять, что каждый символ переводится по таблице ASCII и так хранится в памяти, а последний символ это 0, который указывает на окончание строки.

Массив строк

Создадим два массива строк, которые будут реализованы по-разному и посмотрим на их хранение в памяти.

```
#include <stdio.h>

int main(void)
{
   char arr_1[][9] = {"January", "February", "March"};
   char *arr_2[] = {"January", "February", "March"};
   return 0;
}
```

Для дампа строк, нужно знать их размер, посчитаем их аналитически, а потом проверим при помощи команды **sizeof**. Первый массив предятсавляет из себя двумерный массив из 3 элементов, каждый из которых имеет длинну 9, значит размер всего массива будет равен 9байт * 3 = 27 байт, проверив при помощи команды **sizeof** убеждаемся в праильности расчетов.
Второй массив представляет из себя массив указателей на строки. Память на этот массив выделяется следующим образом: берется размер каждого элемента и суммируются их размеры. В данном примере массив занимает (8байт + 9байт + 6байт) = 23байта. Проверив при помощи команды **sizeof** убеждаемся в правильноси наших расчетов.

Массив строк как матрица:

```
(gdb) x /27xb arr_1
0x7fffffffdc80: 0x4a    0x61    0x6e    0x75    0x61
0x72    0x79    0x00
0x7fffffffdc88: 0x00    0x46    0x65    0x62    0x72
0x75    0x61    0x72
0x7fffffffdc90: 0x79    0x00    0x4d    0x61    0x72
0x63    0x68    0x00
0x7fffffffdc98: 0x00    0x00    0x00
```

Размер “полезных” данных: 23 байт.
Размер “вспомгательных” данных: 4 байт.
% «вспомогательных» данных по отношению ко всем данным: (4/27) * 100% = 15%

Массив строк как массив указателей:

```
0x7fffffffdc60: 0x04    0x60    0x55    0x55    0x55
0x55    0x00    0x00
0x7fffffffdc68: 0x0c    0x60    0x55    0x55    0x55
0x55    0x00    0x00
0x7fffffffdc70: 0x15    0x60    0x55    0x55    0x55
0x55    0x00    0x00
```

Дамп нулевой строки

```
(gdb) x /8xb arr_2[0]
0x555555556004: 0x4a    0x61    0x6e    0x75    0x61
0x72    0x79    0x00
```

Дамп первой строки

```
(gdb) x /9xb arr_2[1]
0x55555555600c: 0x46    0x65    0x62    0x72    0x75
0x61    0x72    0x79
0x555555556014: 0x00
```

Дамп второй строки

```
(gdb) x /6xb arr_2[2]
0x555555556015: 0x4d    0x61    0x72    0x63    0x68
0x00
```

Размер “полезных” данных: 23 байт.
Размер “вспомгательных” данных: 24 байт.
% «вспомогательных» данных по отношению ко всем данным: (24/(24 + 23)) * 100% = 51%