

Отчет

Задание №3.1

Отладка

Целью работы является умение студента самостоятельно производить трассировку приложения.

1. Задание №1

- 1) Чтобы посмотреть программу в отладчике, необходимо скомпилировать программу с ключом -g. Если попытаться открыть программу в отладчике, не указав данный ключ, то отладчик напишет, что не видит отладочной информации в исполняемом файле (No debugging symbols found in app.exe).
- 2) Чтобы запустить программу под отладчиком необходимо прописать команду “gdb app.exe” с учетом того, что в app.exe содержится отладочная информация. Чтобы досрочно завершить программу необходимо написать “q” или комбинацию клавиш “Ctrl+D”. В случае отсутствия дальнейших точек останова можно написать команду “continue”, в таком случае программа выполнится до конца.
- 3) Для просмотра информации о том, на какой точке останова программа была остановлена, необходимо прописать команду “info breakpoints”. Рядом с названием точки останова будет надпись “breakpoint already hit 1 time”, а также строка кода программы и её номер.
- 4) Посмотреть значение переменной можно с помощью команды “print {назв. перем.}”. Чтобы изменить значение переменной необходимо прописать команду “set var {назв. перем.}={значение}”.
- 5) Чтобы выполнить программу в пошаговом режиме, существуют 4 команды:
 - Step – выполняет текущую строку и останавливается на следующем операторе для выполнения (если строка – функция, то step останавливается в её начале).
 - Next - выполняет текущую строку и останавливается на следующем операторе для выполнения (если строка – функция, то next её выполняет и останавливается на следующем операторе).
 - Continue – продолжает обычное выполнение программы до следующей точки останова.
 - Finish – выполняет команды next без остановки, пока не достигнет конца текущей функции.

- 6) Последовательность вызова функций покажет команда “bt” (работает на основе текущего кадра стека).
- 7) Чтобы установить точку останова в программе необходимо прописать команду “break {номер строки}”. Программа остановится на строке с указанным номером.
- 8) Команда “tbreak” создает временную точку останова. Точка останова считается временной, если в ходе отладки программы отладчик останавливается на данной точке только 1 раз.
- 9) Выключить/включить точку останова – команда “disable/enable [номер|диапазон]”. Для игнорирования нескольких срабатываний используем команду “ignore {номер} {кол_во итераций}”.
- 10) Чтобы задать условие на точке останова необходимо прописать команду “break {номер строки} if {условие}”. Если необходимо добавить условие к существующей точке останова – использует команду “condition {номер} {условие}”.
- 11) Точка останова останавливает программу всякий раз, когда её выполнение достигает определенной точки. Точка наблюдения – это специальная точка останова, которая останавливает программу при изменении значения выражения.
- 12) Поскольку точка наблюдения выдает старое значение переменной и новое, то логично её использовать в сложном выражении, в котором вы могли забыть приоритет операций языка Си.
- 13) Содержимое области памяти можно посмотреть с помощью команды “x [/nfu] [адрес]”, где n – сколько единиц памяти должно быть выведено, f – спецификатор формата, u – размер выводимой единицы памяти.

2. Задание №2

Тип	Ubuntu 22.04.01 LTS (размер [Б])	Windows 11 (размер [Б])
char	1	1
int	4	4
unsigned	4	4
long long	8	8
short	2	2
int32_t	4	4
int64_t	8	8

3. Задание №3

Заданы переменные a1, a2, a3, a4 соответствующих типов char, int, unsigned, long long. Значения переменных равно 100:

```
(gdb) x /1xb &a1
0x7fffffffdf2f: 0x64

(gdb) x /4xb &a2
0x7fffffffdf30: 0x64 0x00 0x00 0x00

(gdb) x /4xb &a3
0x7fffffffdf34: 0x64 0x00 0x00 0x00

(gdb) x /8xb &a4
0x7fffffffdf38: 0x64 0x00 0x00 0x00 0x00 0x00 0x00 0x00

(gdb) info locals
a1 = 100 'd'
a2 = 100
a3 = 100
a4 = 100
```

Значения переменных равно -100:

```
(gdb) x /1xb &a1
0x7fffffffdf2f: 0x9c

(gdb) x /4xb &a2
0x7fffffffdf30: 0x9c 0xff 0xff 0xff

(gdb) x /4xb &a3
0x7fffffffdf34: 0x9c 0xff 0xff 0xff

(gdb) x /8xb &a4
0x7fffffffdf38: 0x9c 0xff 0xff 0xff 0xff 0xff 0xff 0xff

(gdb) info locals
a1 = -100 '\234'
a2 = -100
a3 = 4294967196
a4 = -100
```

Вывод: все типы, кроме unsigned, хранят знак (unsigned принимает только положительные значения).

4. Задание №4

Код программы:

```
#include <stdio.h>
#include <stdlib.h>
#include <inttypes.h>
#define OK 0
int main(void)
{
    int a[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    int *pn = a;
    printf("adress - %p, value - %d\n", pn, *pn);
    (*pn)++;
    printf("adress - %p, value - %d\n", pn, *pn);
    pn++;
    printf("adress - %p, value - %d\n", pn, *pn);
    return OK;
}
```

Вывод:

```
ivan@ivan-VirtualBox:~/Рабочий стол/study/sem-2$ ./app.exe
adress - 0x7ffdde0c47d0, value - 1
adress - 0x7ffdde0c47d0, value - 2
adress - 0x7ffdde0c47d4, value - 2
```

Убедимся с помощью gdb, что pn – указатель:

```
adress - 0x7ffffffdf10, value - 1
Breakpoint 1, main () at main.c:12
12 (*pn)++;
(gdb) x /10uw a
0x7ffffffdf10: 1 2 3 4
0x7ffffffdf20: 5 6 7 8
0x7ffffffdf30: 9 10
(gdb) x /10xg a
```

```
0x7fffffffdf10: 0x00000000200000001 0x00000000400000003
0x7fffffffdf20: 0x00000000600000005 0x00000000800000007
0x7fffffffdf30: 0x00000000a00000009 0x9b151030f44e2100
0x7fffffffdf40: 0x00000000000000001 0x00007ffffc29d90
0x7fffffffdf50: 0x00000000000000000 0x000055555555169
```

(gdb) cont

Continuing.

adrees - 0x7fffffffdf10, value - 2

Breakpoint 2, main () at main.c:14

14 pn++;

(gdb) x /10uw a

```
0x7fffffffdf10: 2 2 3 4
```

```
0x7fffffffdf20: 5 6 7 8
```

```
0x7fffffffdf30: 9 10
```

(gdb) x /10xg a

```
0x7fffffffdf10: 0x00000000200000002 0x00000000400000003
```

```
0x7fffffffdf20: 0x00000000600000005 0x00000000800000007
```

```
0x7fffffffdf30: 0x00000000a00000009 0x9b151030f44e2100
```

```
0x7fffffffdf40: 0x00000000000000001 0x00007ffffc29d90
```

```
0x7fffffffdf50: 0x00000000000000000 0x000055555555169
```

(gdb) cont

Continuing.

adrees - 0x7fffffffdf14, value - 2

Breakpoint 3, main () at main.c:16

16 return OK;

(gdb) x /10uw a

```
0x7fffffffdf10: 2 2 3 4
```

```
0x7fffffffdf20: 5 6 7 8
```

```
0x7fffffffdf30: 9 10
```

(gdb) x /10xg a

```
0x7fffffffdf10: 0x00000000200000002 0x00000000400000003
```

```
0x7fffffffdf20: 0x00000000600000005 0x00000000800000007
```

0x7fffffffdf30: 0x00000000a00000009 0x9b151030f44e2100

0x7fffffffdf40: 0x00000000000000001 0x00007ffff7c29d90

0x7fffffffdf50: 0x00000000000000000 0x00005555555555169

5. Задание №5

Точка наблюдения нужна тогда, когда переменная изменяется неявно. Дана программа:

```
#include <stdio.h>
#include <stdlib.h>
#include <inttypes.h>
#define OK 0
void func(void* a)
{
    *(int*)a = 30;
}
int main(void)
{
    short a = 10;
    short b = 20;
    func(&a);
    printf("a=%d, b=%d\n", a, b);
    return OK;
}
```

Подразумевается, что программа изменит значение переменной *a*, а значение переменной *b* останется прежним. Однако, запустив программу получаем следующее:

```
ivan@ivan-VirtualBox:~/Рабочий стол/study/sem-2$ ./app.exe
```

```
a=30, b=0
```

Значение переменной *b* почему-то изменилось на 0, хотя нигде не прописано действий с ней. Поставим на неё точку наблюдения:

```
(gdb) watch b
```

```
Hardware watchpoint 2: b
```

```

(gdb) cont
Continuing.
Hardware watchpoint 2: b
Old value = 0
New value = 20
main () at main.c:16
16 func(&a);
(gdb) cont
Continuing.
Hardware watchpoint 2: b
Old value = 20
New value = 0
func (a=0x7ffffffdf34) at main.c:10
10 }
(gdb) cont
Continuing.
a=30, b=0

```

Первая остановка отвечает за инициализацию переменной `b` и присвоения числа 20. Вторая остановка присваивает значение 0. Мы видим, что это происходит в функции `func`, значит в ней и кроется ошибка. Для её решения необходимо поменять в функции тип `int` на тип `short`.

6. Задание №6

Ниже приведена таблица команд `gdb` и соответствующих команд `qt creator`:

Смысл команды	GDB	QT CREATOR
Начать отладку	<code>gdb app.exe</code>	F5 или “Отладка > Начать отладку”
Поставить точку останова	<code>break 17</code>	Навестись на поле слева от строки 17 и нажать ЛКМ (или нажать F9)
Удалить точку останова	<code>delete {номер}</code>	Щелкнуть на красную точку слева от строки или “Точки останова > Удалить точку останова”

Остановить отладку	q	SHIFT+F5
Выполняет текущую строку (не учитывая функцию)	step	F11
Выполняет текущую строку (учитывая функцию)	next	F10
Продолжить отладку	continue	F9 или “Отладка > Продолжить”
Стек	bt	“Отладка > Виды > Стек”
Значение переменной	print {перем.}	В виде “Локальные и наблюдаемые переменные” можно посмотреть информацию

Список литературы:

1. Курс “Проектно-технологическая практика (знакомство с Linux)”:
<https://e-learning.bmstu.ru/iu7/course/view.php?id=76>
2. Курс “Проектно-технологическая практика (тестирование, отладка и профилирование ПО)”:
<https://e-learning.bmstu.ru/iu7/course/view.php?id=73>
3. Официальная документация gcc:
<https://gcc.gnu.org/onlinedocs/gcc/Invoking-GCC.html>
4. Документация “Отладка с помощью GDB”
<http://linux.yaroslavl.ru/docs/altlinux/doc-gnu/gdb/gdb.html>
5. Документация GDB “Остановка и продолжение остановки”
https://www.opennet.ru/docs/RUS/gdb/gdb_6.html
6. Документация GDB “Изменение исполнения”
https://www.opennet.ru/docs/RUS/gdb/gdb_12.html
7. Документация GDB “Исследование данных”

https://www.opennet.ru/docs/RUS/gdb/gdb_9.html

8. Документация “Отладка с помощью QT Creator”

<http://doc.crossplatform.ru/qtcreator/1.2.1/creator-debugging.html>

9. Документация “Функция printf и форматы вывода”

<https://kaf401.rloc.ru/Informatics/formats.htm>

10. Документация ‘x-command’

<https://visualgdb.com/gdbreference/commands/x>