**Задание №3.1 ПТП**

**Отладка**

**Ответы на вопросы:**

1. Ключи компиляции для использования GDB

* Чтобы программа могла быть отлажена в GDB, её нужно скомпилировать с ключом -g, который добавляет отладочную информацию (номера строк, имена переменных и функций) в исполняемый файл.

2. Запуск программы в GDB и досрочное завершение

* Запуск программы:

gdb ./app.exe

* Досрочное завершение:

kill­ – завершит выполнение программы, но оставит сессию GDB активной

quit – для выхода из GDB

3. Определение текущего места остановки

* where или bt — показывают стек вызовов
* list — отображает исходный код вокруг текущей строки
* frame — выводит текущий кадр стека

4. Просмотр и изменение переменных

* Просмотр:  
  print var или p var
* Изменение:  
  set var = значение var

5. Пошаговое выполнение

* step (s) — выполняет следующую строку кода, входит в вызовы функций
* next (n) — выполняет следующую строку кода, пропускает вызовы функций
* stepi и nexti — аналоги для инструкций ассемблера

6. Определение цепочки вызовов функций

Команда backtrace (или bt) отображает стек вызовов — последовательность функций, которая привела к текущей точке останова

7. Установка точек останова

* По имени функции:  
  break (b) func\_name
* По номеру строки:  
  break (b) main.c:15
* По адресу:  
  break (b) \*0x0004xf

8. Временная точка останова

Временная точка (tbreak) автоматически удаляется после первого срабатывания.

9. Управление точками останова

* Включить/выключить точку:  
  disable 1 (отключает точку с номером 1),  
  enable 1 (включает её).
* Пропуск срабатываний:  
  ignore 1 3 — пропустить 3 срабатывания точки №1.

10. Условие остановки

Используйте condition <номер\_точки> <условие>.  
Пример:

condition 1 x > 100

Точка №1 сработает только при x > 100.

11. Разница между точкой остановки и точкой наблюделния

* Точки останова (breakpoints) останавливают выполнение при достижении определённого места в коде.
* Точки наблюдения (watchpoints) останавливают выполнение при изменении значения переменной или выражения.

12. Пример использования watchpoints

Предположим, переменная counter изменяется неожиданно.

Установите:

watch counter

GDB остановит выполнение при любом изменении counter, что поможет найти место ошибки

13. Просмотр памяти

Команда x (examine) позволяет просматривать память.

x/[количество][формат][размер] адрес

Примеры:

* x/4xw &var — вывести 4 слова (4 байта) в hex по адресу var.
* x/10c ptr — вывести 10 символов по адресу ptr.

Форматы:

* x — шестнадцатеричный,
* d — десятичный,
* s — строка,
* c — символ.

**Задание №2**

|  |  |  |
| --- | --- | --- |
| **Тип** | **Размер (байты) в Windows 11 (MinGW 11.0.0)** | **Размер (байты) в Ubuntu (GCC 13.3.0)** |
| char | 1 | 1 |
| int | 4 | 4 |
| unsigned | 4 | 4 |
| short int | 2 | 2 |
| long int | 4 | 8 |
| long long | 8 | 8 |
| int32\_t | 4 | 4 |
| int64\_t | 8 | 8 |

**Задание №3**

|  |  |
| --- | --- |
| **Переменная** | **Представление в памяти** |
| char c1 = 'a'; | 0x7fffffffdc0f:0x61 |
| char c2 = -100; | 0x7fffffffdc0e:0x9c |
| int i1 = 5; | 0x7fffffffdc10:05 00 00 00 |
| int i2 = -5; | 0x7fffffffdc14:fb ff ff ff |
| unsigned u1 = 10; | 0x7fffffffdc18:0a 00 00 00 |
| long long ll1 = 255; | 0x7fffffffdc20:ff 00 00 00 00 00 00 00 |
| long long ll2 = -255; | 0x7fffffffdc28:01 ff ff ff ff ff ff ff |

Подробные пояснения для каждой переменной

1. char c1 = 'a';

* ASCII-код символа 'a' — 0x61

В памяти: один байт 0x61

2. char c2 = -100;

* Прямой код 100 → 0x64 (бинарно: 01100100)
* Инверсия битов → 10011011 (0x9B)
* Добавляем 1 → 0x9C (10011100)

В памяти: 0x9c

3. int i1 = 5;

* Прямой код 5 → 0x00000005 (32 бита)

Little-endian: 05 00 00 00

4. int i2 = -5;

* Прямой код 5 → 0x00000005
* Инверсия битов → 0xFFFFFFFA
* Добавляем 1 → 0xFFFFFFFB

Little-endian: fb ff ff ff

5. long long ll2 = -255;

* Дополнительный код для -255 (64 бита):
* Прямой код 255 → 0x00000000000000FF
* Инверсия битов → 0xFFFFFFFFFFFFFF00
* Добавляем 1 → 0xFFFFFFFFFFFFFF01

Little-endian: 01 ff ff ff ff ff ff ff

**Задание №3**

Листинг тестового кода:

#include <stdio.h>

int main()

{

    int arr[] = {10, 20, 30};

    int \*ptr = arr;

    return 0;

}

На моей системе int занимает 4 байта

Вывод всего массива:

(gdb) x /12xb &arr

0x7fffffffdd6c: 0x0a 0x00 0x00 0x00 0x14 0x00 0x00 0x00

0x7fffffffdd74: 0x1e 0x00 0x00 0x00

Вывод поэлементно:

(gdb) x /4xb ptr

0x7fffffffdd6c: 0x0a 0x00 0x00 0x00

(gdb) x /4xb ptr + 1

0x7fffffffdd70: 0x14 0x00 0x00 0x00

(gdb) x /4xb ptr + 2

0x7fffffffdd74: 0x1e 0x00 0x00 0x00