# Особенности использования отладчика GDB.

**Цель работы.**

Ознакомиться с особенностями использования отладчика GDB.

Важным этапом разработки программного обеспечения является отладка написанной программы. Рассмотрим данный процесс при использовании программы GDB (GNU Debugger), входящей в набор утилит GCC. Дальнейшее описание предполагает работу в операционной системе семейства Windows, с установленным IDE Dev-Cpp. Будет использован набор компиляторов и утилит GCC, входящий в состав IDE Dev-Cpp. Путь установки для Dev-Cpp предполагается следующий: **C:\dev-cpp.** Рабочий каталог для хранения исходных текстов - **c:\users**. При использовании других путей к рабочему каталогу и Dev-Cpp, необходимо соответствующим образом отредактировать приведённые команды. При использовании операционной системы из семейства Linux не требуется устанавливать IDE Dev-Cpp и указывать путь к программам, входящим в состав GCC.

**Порядок выполнения работы.**

**Подготовка исходного текста программы к использованию отладчика.**

Для демонстрации процесса отладки, создадим файл **gdbtest.c** со следующим исходным текстом программы:

```
#include <stdio.h>

int main()
{
      int i=0;
      for(i=1;i<5;i++);
      {
            printf("2i=%d\n",2*i);
      }
      return 0;
}
```

Предполагаемым результатом работы данной программы должен быть вывод на экран чисел 2,4,6,8, однако из-за допущенной ошибки, на экран выводится только число 10.

Теперь скомпилируем этот файл командой **C:\dev-cpp\bin\gcc -g c:\users\gdbtest.c -o c:\users\gdbtest.exe**. Параметр **-g** отвечает за добавление в исполняемый файл отладочной информации.

**Использование отладчика**

Следующим шагом будет запуск отладки программы командой **C:\dev-cpp\bin\gdb c:\users\gdbtest.exe**. После её выполнения, появится информация об отладчике и приглашение к вводу команд **(gdb)**. После приглашения (gdb) можно вводить различные команды. Многие команды можно сокращать до одной-двух букв. Можно использовать клавишу <Tab> для завершения имени команды, переменной или функции. Простое нажатие клавиши <Enter> без указания команды повторяет последнюю ранее введенную команду. Для завершения работы GDB введите команду **quit** (сокращается до q). Для начала проверим работоспособность программы командой **run**. Результат работы программы будет выведен на экран.

Теперь попробуем пошаговое выполнение программы. Для начала командой **break 4** или **b 4** установим точку останова выполнения на 4й строчке исходного текста программы. После выполнения команды **run** или **r**, программа остановится и выведет на экран текущую строчку исходного кода (в данном случае for(i=1;i<5;i++); ). Теперь можно проверить значение переменной i с помощью команды **print i** или **p i.** После этого командой **step** или **s** перейдём на следующую строчку исходного текста и вновь проверим значение переменной **i**. Мы видим что её значение отличается от ожидаемого (5 вместо 1), что должно подсказать нам строчку, в которой допущена ошибка. Очевидно, что в нашем случае ошибка допущена в строке с началом цикла for - неверно установленная точка с запятой привела к отсутствию тела цикла. Исправим эту ошибку и заново откомпилируем программу. Предварительно следует завершить работу отладчика командой **quit** или **q**.

Вновь запустим отладчик для исправленной версии программы, и установим точку останова на 4-й строке исходного текста. Вместо того, чтобы на каждом шаге работы программы проверять значение переменной **i**, добавим её в наблюдаемые командой **display i.** Теперь используя команду **step** пройдём до конца выполнения программы, проверяя соответствие значения переменной **i** ожидаемому.

### Основные команды отладчика:

**help (h)** - выводит справку по командам, при использовании без параметра выводит список разделов справки, информация по каждому разделу выводится командой **help <имя раздела>**;

**run (r)** начинает исполнение программы до первой точки останова или возникновения ошибки вроде Floating point exception или Segmentation fault. Для перенаправления ввода-вывода команды можно использовать команду **run <input_file >output_file**;

**continue (c)** продолжить выполнение программы до следующей точки останова или ошибки;

**step (s)** Продолжить выполнение программы пока управление не достигнет следующей строки исходного текста. При этом происходит вход в функцию, то есть при вызове каждой функции программа также останавливается;

**next (n)** Работает аналогично команде **step**, но при этом функции вызываются без остановки;

**finish** Продолжить выполнение программы до возврата из текущей функции, напечатать возвращаемое значение;

**print <expr> (p)** Отобразить значение выражения **expr**. Допускается использование практически любого выражения языка C/C++, в том числе и вызовы функций. Например: **p a, p**

**sqrt(n), p (a&b)>>3** Если имеется динамический массив (например, **int * p=new int[10]**), то при выводе его на экран (**print p**) будет напечатано значение указателя, то есть адрес в памяти компьютера. Если хочется вывести весь массив (или его часть), то необходимо воспользоваться оператором создания виртуального массива **@**, левый операнд которого - первый элемент запрашиваемого массива, а правый - длина желаемого массива. Например, **print *p@10** или **print p[0]@10**;

**display <expr>** Добавить выражение **expr** к списку автоматического отображения. Каждое такое выражение получает свой номер;

**delete display n** Удалить выражение с номером **n** из списка автоматического отображения;

**disable display n** Отключить отображение элемента с номером **n** из списка, не удаляя его;

**enable display n** Включить отображение элемента **n** из списка, отключенного ранее;

**display** Напечатать значения всех автоматически отображаемых выражений, как это происходит при остановке программы;

**info display** Вывести список автоматически отображаемых выражений;

В произвольном месте программы можно поставить точку останова, при достижении которой команда остановится. При этом можно исследовать значения переменных, продолжить выполнение программы при помощи команд **c, s, n** или выполнить другие команды GDB.

**break n (b)** Установить точку останова на строке исходного текста с номером **n**. Все точки останова имеют свои номера, определяемые при их установке, если **n** не указана, точка останова устанавливается на текущей строке исходного текста;

**break <function>** Установить точку останова на функции **function**. Исполнение программы остановится при вызове этой функции. Распространенный пример: **break main** для того, чтобы можно было пошагово исполнять программу с самого начала;

**tbreak** ... Установка разовой точки останова. Работает аналогично **break**, но точка останова автоматически удаляется при достижении ее;

**break <n or function> if <cond>** Установка условной точки останова. Выполнение останавливается на указанной строке или функции, если выполнено условие **cond**. Пример: **break 120 if a==15**;

**clear <function>** или **clear n** Удалить точку останова на функции **function** или в строке с номером **n**;

**delete <диапазон>** (**d**) Удалить точки останова с номерами из указанного диапазона. Например, **delete 1-5** или **delete 4**. Если диапазон не задан, удаляются все точки останова;

**disable <диапазон>** Отключить точки из заданного диапазона;

**enable <диапазон>** Включить точки из заданного диапазона;

**enable once <диапазон>** Точки из заданного диапазона включаются до первого срабатывания, после чего отключаются;

**enable delete <диапазон>** Точки из заданного диапазона включаются до первого срабатывания, после чего удаляются;

**condition n <cond>** Превращение точки останова с номером **n** в условную с условием **cond**;

**condition n** Превращение точки останова с номером **n** в безусловную;

**info break** Вывести информацию обо всех имеющихся точках останова;

**watch <expr>** Установить точку наблюдения на выражении <**expr>.** Исполнение программы останавливается, когда значение **expr** сохраняется программой и его величина изменяется. Пример: **watch a**;

**rwatch <expr>** Установить точку наблюдения, которая остановит программу, когда выражение **expr** считывается программой;

**awatch <expr>** Установить точку наблюдения, которая остановит программу, если выражение **expr** считывается или сохраняется;

**info watch** Вывести информацию о всех точках наблюдения;

Для вывода исходного текста программы используется команда list (l) в различных вариантах:

**list n** Вывести строки исходного кода, вокруг строки с номером **n;**

**list <function>** Вывести начало кода функции **function;**

**list n,m** Вывести строки с номерами от **n** до **m;**

**list n**, Вывесть строки, начиная с **n**;

**list ,m** Вывести строки до номера **m**;

**list** Вывести еще немного строк. По умолчанию выводится 10 следующих строк, после ранее выведенных. Это значение можно изменить командой **set listsize <число>**;

Всякий раз, когда вызывается функция, информация о вызове сохраняется в стеке. Информация сохраняется в блоке данных, называемом кадром стека.

**where** Вывести содержимое стека;

**frame n** Переключиться в кадр с номером **n**. При этом можно просматривать значения локальных переменных в контексте этого кадра;

**up n** Переместиться по стеку вверх на **n** кадров. По умолчанию **n** равно 1;

**down n** Переместиться по стеку вниз на **n** кадров;

**info frame** Вывести информацию о текущем кадре;

**info args** Вывести информацию об аргументах текущего кадра;

**info locals** Вывести информацию о всех локальных переменных текущего кадра.

**Задания к лабораторной работе.**

## Часть I

1. Написать программу №1 в соответствии с вариантом при помощи любого текстового редактора.
2. Скомпилировать программу с добавлением в файл отладочной информации.
3. Используя отладчик GDB, проверить значение вычисляемого в цикле выражения на каждом шаге цикла. При использовании нескольких циклов, проверять все значения.
4. В отчёте привести используемые команды отладчика и полученный результат

## Часть II

1. Написать программу №2 в соответствии с вариантом при помощи любого текстового редактора.
2. Для ввода и вывода строки использовать отдельные функции.
3. Скомпилировать программу с добавлением в файл отладочной информации.
4. Используя отладчик GDB, проверить содержимое стека при входе в функции ввода и вывода строки и выходе из них.
5. В отчёте привести содержимое стека и используемые команды.

## Часть III

1. Написать программу №3 в соответствии с вариантом при помощи любого текстового редактора.
2. Для ввода и вывода строки использовать отдельные функции, помещённые в статическую библиотеку.
3. Скомпилировать программу с добавлением в файл отладочной информации.
4. Используя отладчик GDB, проверить содержимое стека при входе в функции ввода и вывода строки и выходе из них.
5. В отчёте привести содержимое стека и используемые команды.

**Варианты к заданию.**

Вариант 1

1. M и N – числитель и знаменатель обыкновенной дроби. Составить программу, позволяющую сократить эту дробь.

2. Дана строка символов до точки. Выделить в ней все русские буквы, сделав их заглавными.
3. Дана строка символов до точки. Определить, является ли она палиндромом. (Палиндром слева направо и справа налево читается одинаково, например "Леша на полке клопа нашел.") Пробелы и знаки препинания, а также регистр букв не учитываются.

Вариант 2

1. Представить натуральное число N в виде произведения простых сомножителей.
2. Дана строка символов. Удалить из нее все слова нечетной длины. Слова отделяются друг от друга одним пробелом.
3. Дана строка символов. Подсчитать, сколько раз в ней встречается подслово «ABBA».

Вариант 3

1. Составить программу для определения, является ли натуральное число k степенью числа
2. Дана строка символов (введена с клавиатуры), состоящая из цифр от 0 до 9. Составить новую строку из букв от A (соответствует цифре 0) до J (соответствует цифре 9).
3. Дана символьная строка. Заменить в ней все буквы «О» на «Ъ», а буквы «Е» – на случайные символы.

Вариант 4

1. Дано целое $m>1$. Получить наибольшее целое k, при котором $4k<m$.
2. Дана строка символов до точки. Определить, входят ли в состав заданной строки цифры. Сформировать из них новую строку.

3. Дана строка символов до точки. Изменить ее таким образом, чтобы все символы отделялись друг от друга одним пробелом.

Вариант 5

1. Найти наименьший общий делитель трех натуральных чисел (1 будет считаться наименьшим общим делителем только в том случае, когда других общих делителей у заданных чисел нет).
2. Дана строка символов до точки. Записать слова этой строки в обратном порядке (мама мыла раму → раму мыла мама).
3. Дана строка символов до точки. Подсчитать, сколько каких знаков препинания она содержит.

Вариант 6

1. Определить, является ли данное натуральное число N факториалом какого-нибудь числа, если «да», то какого.
2. Дана строка символов. Определить, является ли она правильным скобочным выражением.
3. Дана символьная строка. Проверить, все ли слова после точки начинаются с заглавной буквы. Если нет – исправить.

Вариант 7

1. Найти число Фибоначчи, ближайшее к заданному натуральному числу N.
2. Даны две строки. Составить третью строку из слов, имеющихся в обеих данных строках.
3. Дана строка символов. Выделить и вывести слова, ограниченные пробелом или знаками препинания: запятая, точка, двоеточие, точка с запятой.

Вариант 8

1. С клавиатуры вводится последовательность чисел, признак окончания ввода – ввод 0. Найти максимальное из них.
2. Дана строка символов. Изменить строку – во всех словах, имеющих нечетное количество символов, средний символ удалить.
3. Дана строка символов. Найти в строке самое длинное слово и вывести его на экран.

Вариант 9

1. С клавиатуры вводится последовательность натуральных чисел, признак окончания ввода – ввод 0. Найти все числа, оканчивающиеся на 7.
2. Дана строка символов до точки. Оставить в ней только слова, начинающиеся на буквы «A», «D», «K», «P».
3. Дана строка символов до точки. Удалить из нее все сочетания “ac”.

Вариант 10

1. Составить программу для определения, в каких двузначных числах удвоенная сумма цифр равна их произведению.
2. Дана символьная строка. Оставить в ней только слова, содержащие хотя бы одну букву «A».
3. Дана строка, состоящая из букв и цифр. Преобразовать ее так, чтобы сначала шли буквы, а потом – все цифры исходной строки.

Вариант 11

1. С клавиатуры вводится число N. Определить, может ли оно быть восьмеричным (т.е. состоять только из цифр меньше 8).

2. Дана строка символов, состоящая из букв и цифр. Найти количество букв в данной строке.
3. Дана строка символов. Поменять местами первое и последнее слова.

Вариант 12

1. Дано натуральное число n>10. Составить программу для вычисления значения $y = n\cos x + (n-1)\cos 2x + (n-2)\cos 3x + ...2\cos(n-1)x + \cos nx$.
2. Дана строка символов до точки. Удалить из нее все слова, начинающиеся с буквы «а».
3. Дана строка символов до точки. Удалить из нее все повторные вхождения букв.

Вариант 13

1. Дано натуральное число N. Вычислить произведение $\left(1+\frac{1}{1^2}\right)*\left(1+\frac{1}{2^2}\right)*...*\left(1+\frac{1}{N^2}\right)$.
2. Дана символьная строка. Удалить из нее все символы, не являющиеся буквами.
3. Дана строка, состоящая из букв от A до J. Составить новую строку из цифр от 0 (соответствует цифре A) до 9 (соответствует цифре J).

Вариант 14

1. Составить программу для вычисления значения $Y = \sin 1 + \sin 1.1 + \sin 1.2 + ... + \sin 2$.
2. Дана строка символов. Поменять местами N первых и N последних символов заданной строки.

3. Дана строка, состоящая из букв и цифр. Проверить, является ли данная строка представлением числа в шестнадцатеричной системе счисления.

Вариант 15

1. Даны два натуральных числа X и Y. Составить программу для вычисления суммы кубов всех четных чисел, лежащих в диапазоне [X, Y].
2. Дана строка символов до точки. Инвертировать каждое слово в заданной строке.
3. Дана строка символов до точки. Определить, сколько каких латинских букв в ней содержится.

**Контрольные вопросы.**

1) Что такое GDB?
2) Что такое отладка программного обеспечения?
3) Каким образом должны быть подготовлены программы, чтобы их можно было исследовать с помощью GDB?
4) Что такое «точка останова»? Сколько точек останова можно задать в процессе отладки программы? Можно ли задать условие срабатывания точки останова?
5) Что такое “точка наблюдения” и как их использовать?
6) Что такое стек? Что в него заносится при работе программы?
7) Просмотр стека с помощью GDB.
8) Список автоматического отображения и работа с ним.
9) Просмотр исходного текста программы с помощью GDB.
10) Основные команды GDB.
11) Какую именно информацию необходимо добавить в программу на

этапе компиляции для работы отладчика?

Использована статья “О GCC, компиляции и библиотеках“, расположенная по адресу http://pyviy.blogspot.ru/2010/12/gcc.html

Также использованы материалы из статьи “Отладка программ при помощи GDB”, расположенной по адресу http://server.179.ru/tasks/gdb/

Задачи взяты из методического пособия “Основы программирования на языке Си”, составители:

Арипова Ольга Владимировна, Бузюкина Ольга Александровна