

Отладка

Идея – программы часто не работают так, как того хочет программист. Какие ошибки сложнее всего искать?

1. Runtime errors, приводящие к немедленной остановке программы. SEGFAULT, Double free corruption и т.п
2. Логические ошибки, которые приводят к неверному поведению программы

Есть несколько техник, как искать подобные ошибки.

Способ 1. Отладочный вывод

Основные полезные моменты

1. Все отладочные выводы следует делать в поток stderr
2. При отладке полезно использовать макросы, сообщающие, например, номер строки и имя функции
3. Полезно уметь «отключать» отладочные сообщения

Простой пример отладочных выводов «на коленке» (более подробно этот вопрос рассматривается в статье [Logging with GCC](#)):

```
#include <stdio.h>

#define DEBUG

int main(){
    #ifdef DEBUG
        fprintf(stderr, "DEBUG: %s:%s:%d: %s\n", __FILE__, __func__, __LINE__,
                    "Debug message");
    #endif
    return 0;
}
```

Почитать подробнее

- [Полезные стандартные макросы](#)
- [Статья про простое логгирование \(Logging with GCC\)](#)
- [Использование syslog](#)

Способ 2. gdb + cli

GDB имеет достаточно простой, но мощный командный интерфейс и хорошую справку по нему.

Пример:

```
#include <stdio.h>
#include <stdarg.h>

void get_int(int* val) {
    val = NULL;
    printf("%d\n", *val);
}

int main() {
    printf("Hello!");

    int* a;
    get_int(a);

    return 0;
}
```

При запуске данной программы появляется ошибка segmentation fault, т.е попытка доступа к несуществующей/чужой памяти. Что можно сделать что её найти:

1. Найти строчку, где происходит непосредственно обращение к невалидной памяти
2. Изучить состояние переменных, памяти в тот момент, когда произошла ошибка

Для этого необходимо:

1. Собрать программу с добавлением отладочных данных:

```
gcc -g myprog.c
```

2. Открыть её в отладчике:

```
gdb ./a.out
```

3. Запустить программу командой run:

```
(gdb) run
```

4. Ввести исходные данные, если ваша программа получает какие-то данные на вход. Если требуется перенаправить на вход вашей программе содержимое файла, запустите ее с помощью

```
(gdb) run < input_file.txt
```

Вывод для программы после команды run будет следующий:

```
(gdb) run
Starting program: ./a.out
Program received signal SIGSEGV, Segmentation fault.
0x00005555555555165 in get_int (val=0x0) at main.c:6
```

```
6 printf("%d\n", *val);
```

Теперь вы можете изучить состояние программы, например:

Написать

```
(gdb) where
```

и получить подробный стектрейс, чтобы узнать в каком файле и функции произошла ошибка:

```
(gdb) where
#0  0x0000555555555165 in get_int (val=0x0) at main.c:6
#1  0x00005555555551ac in main () at main.c:13
```

Изучить состояние переменных с помощью команды «p <variable_name>»

```
(gdb) p val
$1 = (int *) 0x0
```

Если вы хотите изучить состояние программы ДО того, как ошибка случится, то можете использовать команду «b» для расстановки точек останова.

https://www.opennet.ru/docs/RUS/gdb/gdb_6.html

Полезные ссылки:

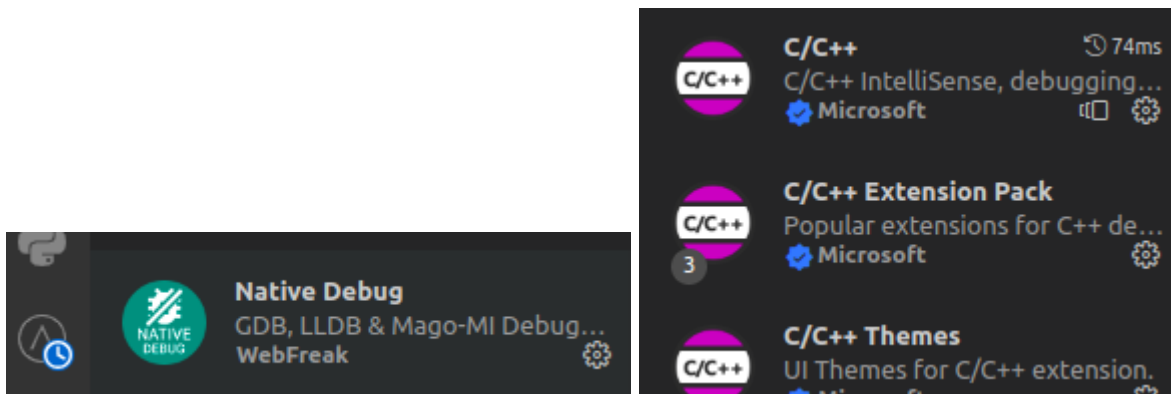
- [Краткий tutorial с примерами отладки ломающихся программ](#)
- [Полезное про массивы и работу с памятью](#)
- [Полная официальная документация](#)

Способ 3. GDB + VSCode

WIP

Чтобы использовать отладчик gdb из IDE VSCode, необходимо:

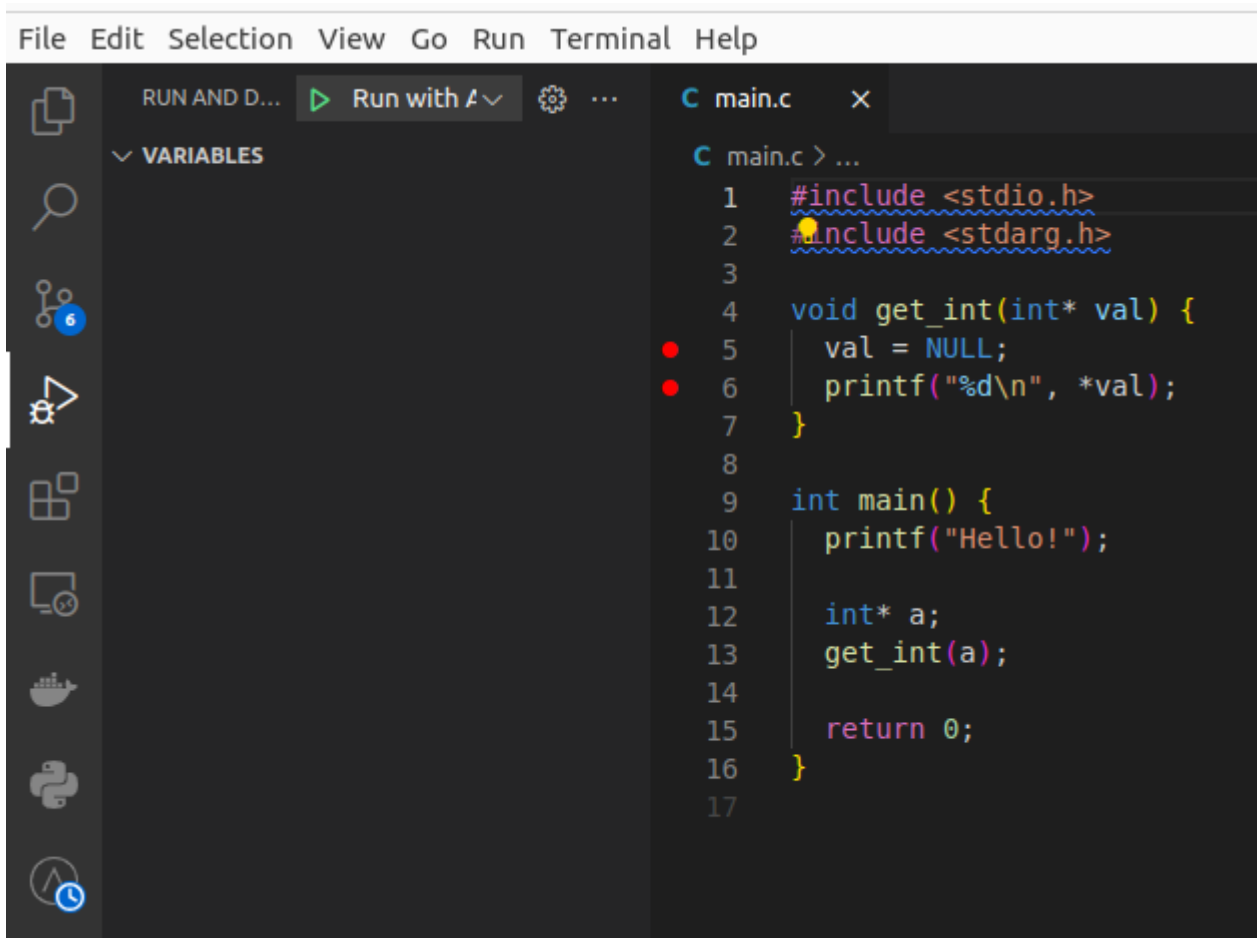
Установить расширения для отладки и работы с языком C:



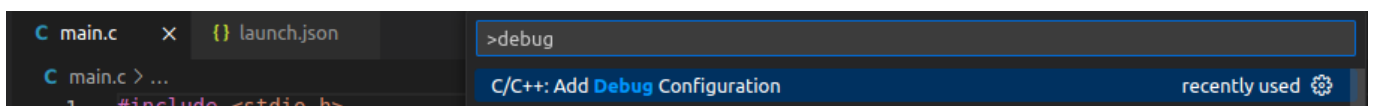
1. Открыть ваш проект/файл в VSCode
2. Расставить точки останова напротив интересующих вас строк кода (нажать слева от

номер строки)

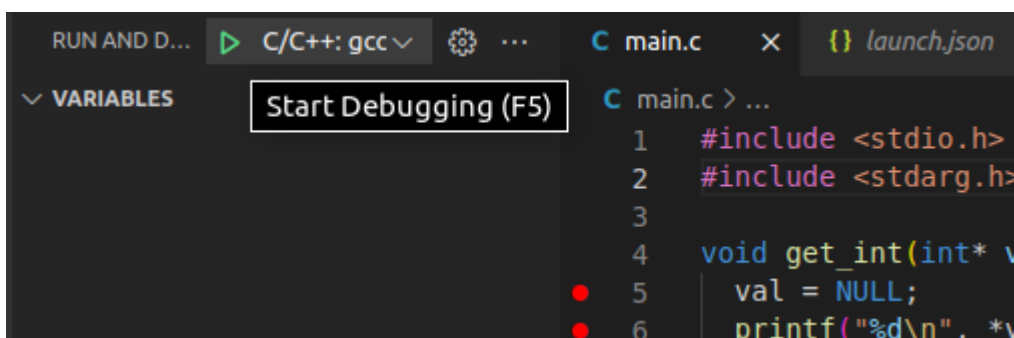
3. Перейти в вкладку «Debug»



Нажать комбинацию клавиш ctrl+shift+P (откроется командная консоль vscode) и написать debug. Выбрать C/C++ debugging

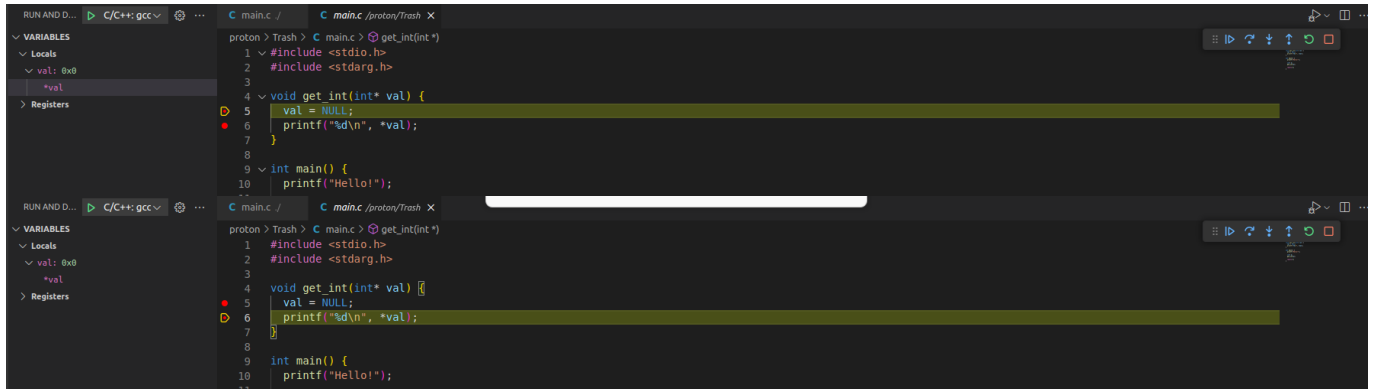


Сохранить файл launch.json. Теперь у вас есть конфигурация для отладки этой программы. Запустите отладку, нажав F5



Теперь в VScode вам доступен интерфейс отладки:

1. Справа панель управления отладкой
2. Слева – состояние памяти и переменных



Архив

gdb + другие ide

Любая среда разработки или даже мощный текстовый редактор обычно предоставляют вам графический интерфейс для использования gdb при отладке своих программ. Обычно он достаточно наглядный и имеет хорошее описание для каждой IDE.

Вы можете самостоятельно найти описание использования отладчика в вашей любимой IDE. Для CLion можно посмотреть эти источники:

- [Debugging in CLion](#)
- [Debugging in CLion on youtube](#)

From:

<https://se.moevm.pro/> - **МОЭВМ Вики** [se.moevm.pro]

Permanent link:

<https://se.moevm.pro/doku.php/courses:programming:debug>

Last update:

