

# GDB - справка

Ссылки:

[Хорошие видео-уроки по GDB, общее время 2:20](#)

[Документация по gdb](#)    [Документация по gdb еще](#)

## Содержание

<b>1</b>	<b>Компиляция</b>	<b>1</b>
<b>2</b>	<b>Запуск</b>	<b>2</b>
<b>3</b>	<b>Основные команды</b>	<b>3</b>
<b>4</b>	<b>Layout</b>	<b>4</b>
<b>5</b>	<b>Переменные</b>	<b>4</b>
<b>6</b>	<b>Просмотр содержимого памяти</b>	<b>4</b>
<b>7</b>	<b>Точки останова</b>	<b>5</b>
<b>8</b>	<b>Точки наблюдения</b>	<b>5</b>
<b>9</b>	<b>Точки перехвата (catchpoints)</b>	<b>6</b>
<b>10</b>	<b>Стек вызовов</b>	<b>6</b>
<b>11</b>	<b>Coredump</b>	<b>6</b>
<b>12</b>	<b>Прочее</b>	<b>7</b>

## 1 Компиляция

Компиляция с флагом `-g`:

```
$gcc -g main.c -o main
```

Проверить наличие отладочной информации:

```
objdump <exefile> | grep debug
```

## 2 Запуск

Запуск программы:

```
gdb ./exefile
```

Запуск без вывода длинного предупреждения в начале:

```
gdb -silent ./exefile
```

Подключение к запущенному процессу по его pid:

```
gdb ./exefile pid
```

Запуск с чтением coredump:

```
gdb -c dump ./exefile
```

Запуск с параметрами:

```
gdb --args ./exefile params
```

Установка параметров после входа в gdb:

```
(gdb) set args params
```

Установка параметров с одновременным запуском:

```
(gdb) r[un] params
```

Установить точки останова и другие действия через командную строку:

```
gdb -ex 'break main' -ex 'info b' -ex 'set print pretty on' ./exefile
```

Запустить сервер gdb для отладки:

```
gdbserver host:port program
```

Присоединиться к gdbserver:

```
target remote host:port
```

### 3 Основные команды

r[un]	Запуск на выполнение
r[un] <arg1> ...	Запуск с параметрами
start	Войти в main и остановиться
b[reak]	Создать точку останова в текущей строке
p[rint] <var>	Вывести значение переменной
x <addr>	Вывести содержимое памяти по адресу
p[type] <var>	Тип значения переменной
h[elp]	Просмотр справки по команде
q[uit]	Выход из программы
backtrace (bt)	Вывести стек вызовов
l[ist]	Вывести код программы
f[rame]	Вывести информацию о текущем фрейме
thread number	Просмотр списка потоков
telescope	Просмотр стека
telescope \$rsp+64	
attach PID	Присоединиться к процессу
detach	Отключиться от процесса

n[ext]	Выполнить следующую строчку без захода в функцию
n[ext] <x>	Выполнить x строчек
s[tep]	Выполнить следующую строчку с заходом в функцию
c[ontinue]	Продолжить выполнение программы
finish	Выйти из функции
u[ntil] <line>	Продолжить выполнение до строки
u[ntil] *func+offset	Продолжить выполнение до строки функции
u[ntil] *address	Продолжить выполнение до адреса

Команды info (i):	
i b[reakpoints]	Вывести список точек останова
i lo[cals]	Вывести значения локальных переменных
i args	Вывести значения аргументов функции
i r[egisters]	Вывести значения регистров
i threads	Список потоков
i file	Просмотреть информацию об архитектуре, секциях
i func[tions]	Получение списка функций
i proc mappings	Распределение виртуальной памяти

## 4 Layout

`tui enable` - Включить отображение кода  
`tui disable` - Отключить отображение кода  
`Ctrl-X + Ctrl-A` - Включить/отключить отображение кода  
`Ctrl-L` - Обновить отображение кода  
`layout src` - Включить отображение кода C  
`layout asm` - Включить отображение asm  
`layout reg` - Включить отображение регистров  
`display <var>` - Добавить в вывод значение переменной  
`undisplay <var>` - Удалить из вывода значение переменной

Выбрать синтаксис ассемблера intel/att:

```
set disassembly-flavor intel
set disassembly-flavor att
```

Получение asm-листинга функции:

```
disas func_name
disas address
```

## 5 Переменные

`p[rint] <var>` - Вывести значение переменной  
`p[rint] <var>=<value>` - Установить значение переменной  
`set var <var>=<value>` - Установить значение переменной  
`p[rint] <var>` - Вывести тип значения переменной  
`i[nfo] lo[cals]` - Вывести значения локальных переменных  
`i[nfo] args` - Вывести значения аргументов текущей функции

## 6 Просмотр содержимого памяти

`x/nfu address`

**n** - количество единиц (1 можно не указывать)

**f** - формат:

- **o** - восьмиричный;
- **x** - шестнадцатиричный;
- **d** - десятичный;
- **f** - число с плавающей запятой;
- **i** - инструкция процессора;
- **c** - символ;

- **s** - строка.

**u** - единица данных:

- **b** - байт;
- **h** - полуслово (два байта);
- **w** - слово (четыре байта);
- **g** - восемь байт;

## 7 Точки останова

Установка точек останова:

- **b[reak]** - на текущей строке
- **b[reak] [filename:]n** - в файле на строке **n**
- **b [filename:]function**
- **b \*function+offset**
- **b \*address**
- **f +-n** - на **n** строк ниже (выше)
- **break arg if condition** - точка останова с условием
- **condition n newcondition**
- **tbreak** - разово установить точку останова
- **i[nfo] breakpoints** - информация обо всех точка останова
- **disable [breakpoints] [n-m]** - деактивировать точки останова
- **enable [breakpoints] [n-m]** - активировать точки останова

Удаление точек останова:

- **clear** - удалить в текущей строке
- **clear [filename:]n**
- **clear [filename:]function**
- **d[ele]te [breakpoints] [n-m]**
- **d[ele]te** - удалить все точки останова

## 8 Точки наблюдения

- **wa[tch] expression** - остановить при изменении
- **rw[at]ch expression** - остановить при чтении
- **aw[at]ch expression** - остановить при чтении или изменении
- **i[nfo] watchpoints** - вывести информацию обо всех точках наблюдения

## 9 Точки перехвата (catchpoints)

Перехват C++ исключений:

```
catch [re]throw [regex]
catch catch [regex]
```

Вызов системных функций:

```
catch syscall write
```

Загрузка/выгрузка .so файлов:

```
catch [un]load [regex]
```

Перехват сигналов:

```
catch signal <signal>
```

## 10 Стек вызовов

Текущий стек вызовов:

```
where
backtrace (bt)
info stack
```

`bt <n>` - только <n> последних фрейма

`i[nfo] f[rame]` - подробная информация о фрейме

`f[rame]` - последний фрейм

`f[rame] 1` - предыдущий фрейм

`up/down` - переключение на один фрейм

`up <n>` - выйти на n фреймов вверх

## 11 Coredump

Запуск с coredump:

```
gdb -c <core> <exe>
```

Просмотр настроек размера coredump (нужен параметр `core file size`):

```
ulimit -a
```

Установить неограниченный размер (после перезагрузки сбрасывается):

```
ulimit -c unlimited
```

Шаблон создания coredump:

```
/proc/sys/kernel/core_pattern
```

Пример шаблона для создания в текущем каталоге:

```
echo "core.%e.%p sudo tee /proc/sys/kernel/core_pattern"
```

Может потребоваться сделать настройки для сервиса apport в  
/.config/apport/settings:

```
[main]  
unpackaged=true
```

Для создания coredump может потребоваться запуск сервиса apport:

```
sudo service apport start
```

Дампы могут создаваться тут:

```
/var/crash
```

Статья по настройке coredump:

Where do I find core dump files, and how do I view and analyze the backtrace (stack trace) in one?

## 12 Прочее

Сдампить участок памяти:

```
dump memory output_file start_addr end_addr
```

Для использования reverse debug и сохранения состояния программы:

```
record
```

После этого станут доступны следующие команды:

```
reverse-step  
reverse-next
```

Просмотр секций ELF файла:

```
readelf -WS <exefile>
```

Прыгнуть на нужную строку:

```
set $pc = 0x4005a5  
set $pc += 10
```

Изменить код программы, установить NOP:

```
set *(unsigned char*)0x5555554009b2 = 0x90  
set *(unsigned char*)0x5555554009b3 = 0x90
```