

# GDB - справка

Ссылки:

Хорошие видео-уроки по GDB, общее время 2:20  
Документация по gdb      Документация по gdb еще

## Содержание

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

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

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

Команды <b>info (i)</b> :	
<b>i b[reakpoints]</b>	Вывести список точек останова
<b>i lo[cals]</b>	Вывести значения локальных переменных
<b>i args</b>	Вывести значения аргументов функции
<b>i r[egisters]</b>	Вывести значения регистров
<b>i threads</b>	Список потоков
<b>i file</b>	Просмотреть информацию об архитектуре, секциях
<b>i func[tions]</b>	Получение списка функций
<b>i proc mappings</b>	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[int] <var> - Вывести значение переменной  
p[int] <var>=<value> - Установить значение переменной  
set var <var>=<value> - Установить значение переменной  
ptype <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 точках 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[elete] [breakpoints] [n-m]**
- **d[elete]** - удалить все точки останова

## 8 Точки наблюдения

- **wa[tch] expression** - остановить при изменении
- **rw[atch] expression** - остановить при чтении
- **aw[atch] 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
```