

Программирование на языке C

ЛЕКЦИЯ 3. ПОДРОБНЕЕ О КОМПИЛЯТОРЕ

Понятие toolchain

Компилятор – лишь жаргонное название, более корректным будет назвать набор инструментов, необходимый для разработки ПО – тулчейном, а процесс превращения исходных кодов в исполнимый файл - сборкой

Мы пользуемся тулчейнами семейства GNU Compiler Collection

Процесс сборки

Исходный код можно писать не только в Эклипсе, а в любом текстовом редакторе

```
hello_world.c
1  #include "stdio.h"
2
3
4  int main()
5  {
6      printf("HELLO WORLD\n");
7      return 0;
8  }
9
```

<- Исходный код программы

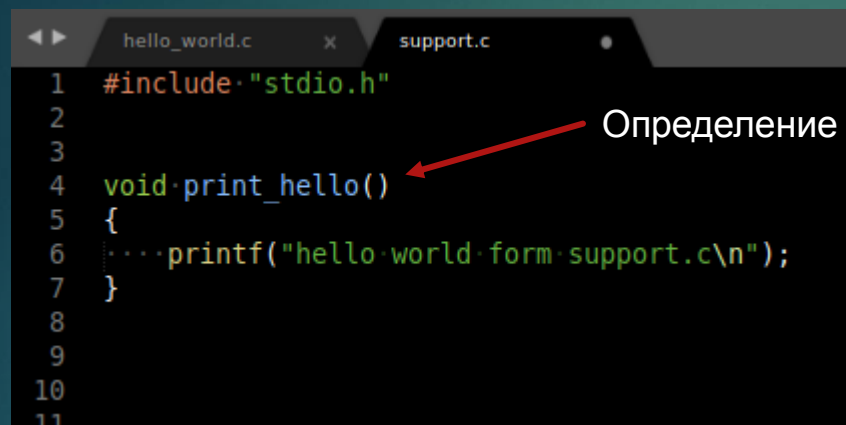
```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ gcc hello_world.c -o hello_world
[snork@snork-pc gcc-test]$
[snork@snork-pc gcc-test]$
[snork@snork-pc gcc-test]$
[snork@snork-pc gcc-test]$ ./hello_world
HELLO WORLD
[snork@snork-pc gcc-test]$
```

«Сборка» программы

Запуск программы

Множество файлов

По мере роста программы, размещать все функции в одном файле становится не удобно



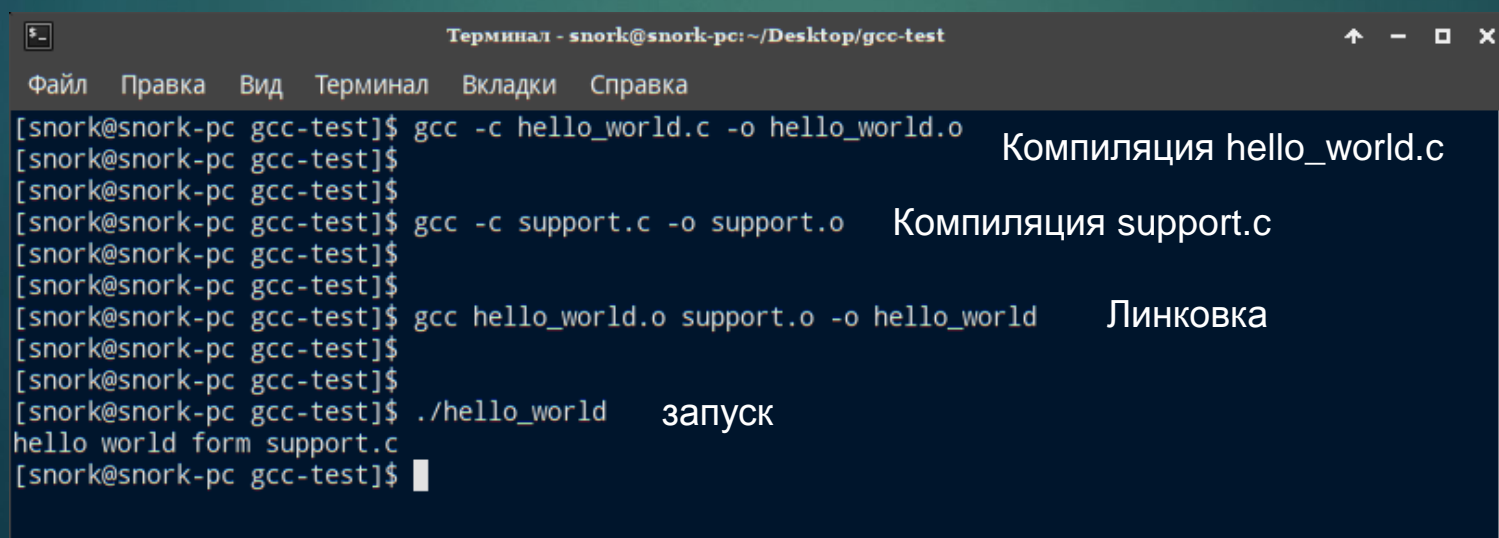
```
1 #include "stdio.h"
2
3
4 void print_hello()
5 {
6     printf("hello world form support.c\n");
7 }
8
9
10
11
```

Определение функции



```
1
2
3 void print_hello();
4
5
6 int main()
7 {
8     print_hello();
9     return 0;
10 }
11
12
13
```

Объявление функции



```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ gcc -c hello_world.c -o hello_world.o
[snork@snork-pc gcc-test]$
[snork@snork-pc gcc-test]$ gcc -c support.c -o support.o
[snork@snork-pc gcc-test]$
[snork@snork-pc gcc-test]$ gcc hello_world.o support.o -o hello_world
[snork@snork-pc gcc-test]$
[snork@snork-pc gcc-test]$ ./hello_world
hello world form support.c
[snork@snork-pc gcc-test]$
```

Компиляция hello_world.c

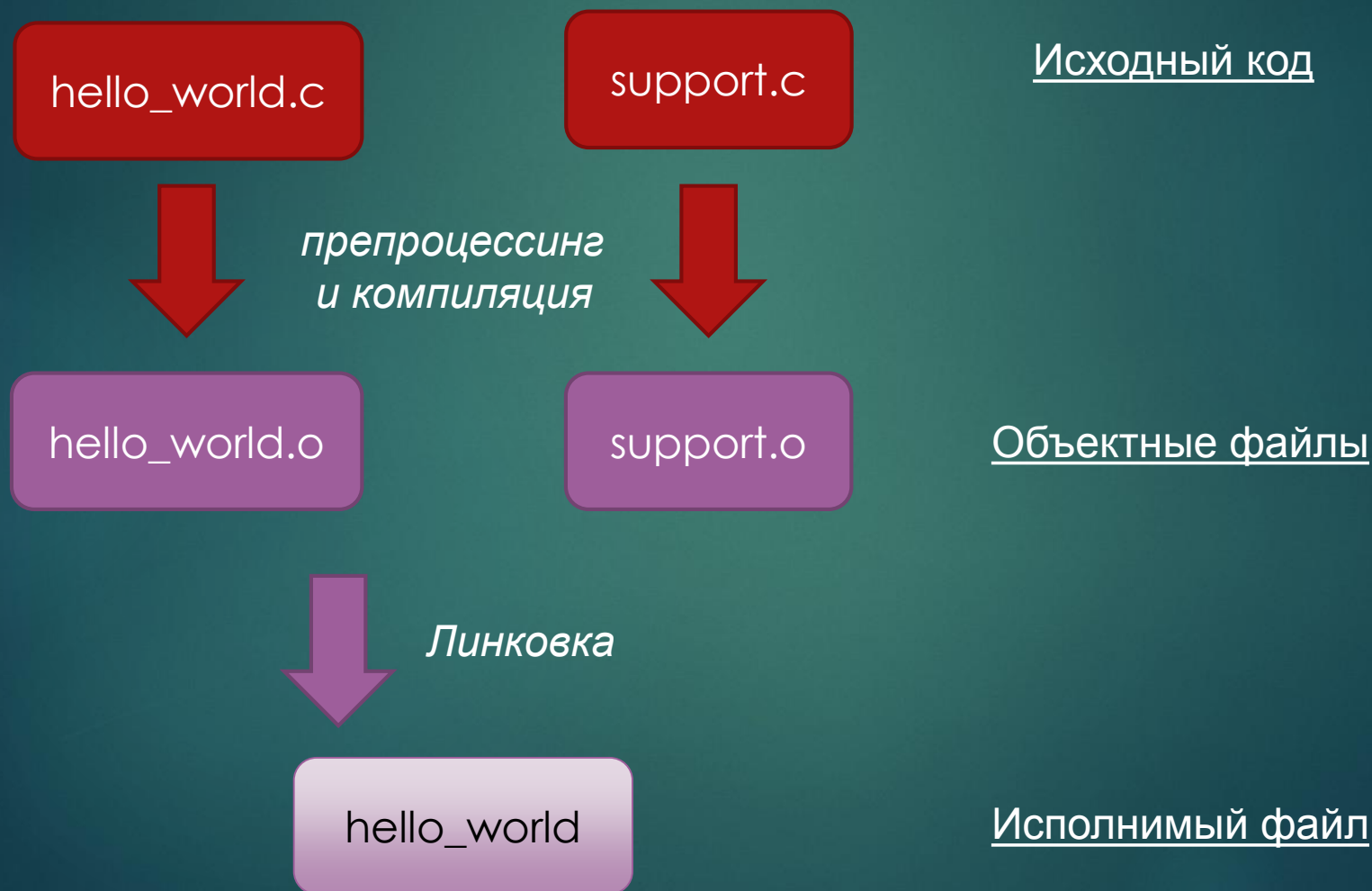
Компиляция support.c

Линковка

запуск

Множество файлов

Тоже самое в виде схемы



Переменные из других файлов

По мере роста программы, размещать все функции в одном файле становится не удобно

```
hello_world.c x support.c
1 #include "stdio.h"
2
3
4 int my_global_variable = 0;
5
6
7 void print_global_variable()
8 {
9     printf("global variable value = %d\n",
10         my_global_variable);
11 }
12
```

Определение переменной

Определение функции

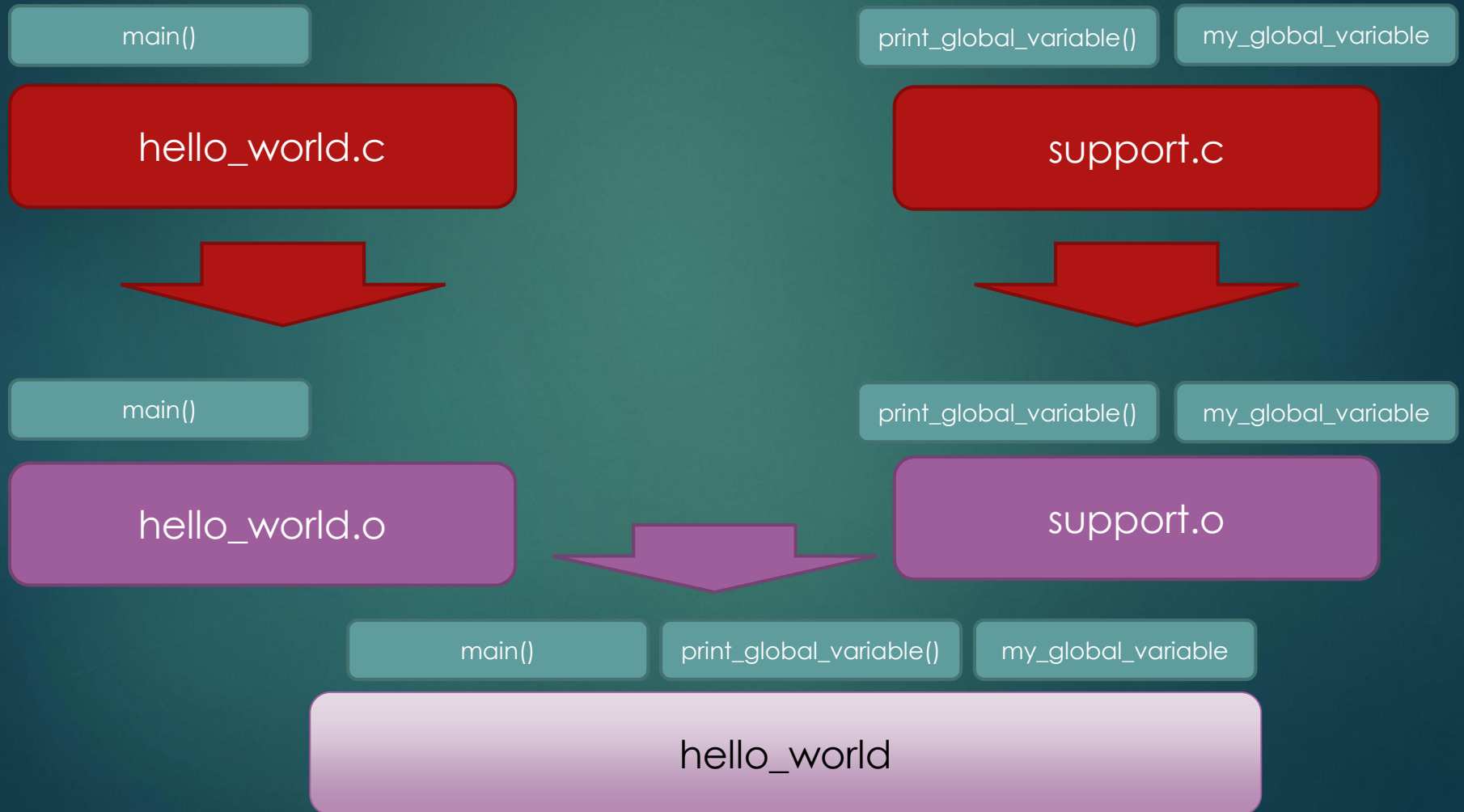
```
hello_world.c x support.c
1
2 extern int my_global_variable;
3 extern void print_global_variable();
4
5
6 int main()
7 {
8     print_global_variable();
9     my_global_variable = 100;
10    print_global_variable();
11    return 0;
12 }
13
14
```

Объявление переменной и функции

```
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ ./hello_world
global variable value = 0
global variable value = 100
[snork@snork-pc gcc-test]$
```

Множество файлов

Тоже самое в виде схемы



ОШИБКИ ЛИНКОВКИ

1. Функция (или переменная) определена дважды

```
main.c
1
2
3 int main()
4 {
5     return 0;
6 }
```

```
a.c
1 #include "stdio.h"
2
3
4 void write()
5 {
6     printf("write in a.c called\n");
7 }
```

```
b.c
1 #include "stdio.h"
2
3 void write(char *data, int data_size)
4 {
5     printf("write from b.c called "
6     "with args %p %d\n", data, data_size)
7 }
```

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ gcc -c a.c -o a.o
[snork@snork-pc gcc-test]$ gcc -c b.c -o b.o
[snork@snork-pc gcc-test]$ gcc -c main.c -o main.o
[snork@snork-pc gcc-test]$ gcc a.o b.o main.o -o main
b.o: In function 'write':
b.c:(.text+0x0): multiple definition of 'write'
a.o:a.c:(.text+0x0): first defined here
collect2: ошибка: выполнение ld завершилось с кодом возврата 1
[snork@snork-pc gcc-test]$
```


ОШИБКИ ЛИНКОВКИ

2. Функция (или переменная) объявлена, но не определена

```
main.c
1
2
3 void do_something();
4
5
6 int main()
7 {
8     ...do_something();
9     ...return 0;
10 }
11
12
13
14
```

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ gcc -c main.c -o main.o
[snork@snork-pc gcc-test]$ gcc main.o -o main
main.o: In function `main':
main.c:(.text+0xa): undefined reference to `do_something'
collect2: ошибка: выполнение ld завершилось с кодом возврата 1
[snork@snork-pc gcc-test]$
```

ОШИБКИ ЛИНКОВКИ

3. Ошибка в объявлении

```
File Edit Selection Find View Go
main.c x a.c
1
2
3 void do_something();
4
5
6 int main()
7 {
8     ...do_something();
9     ...return 0;
10 }
11
12
13
```

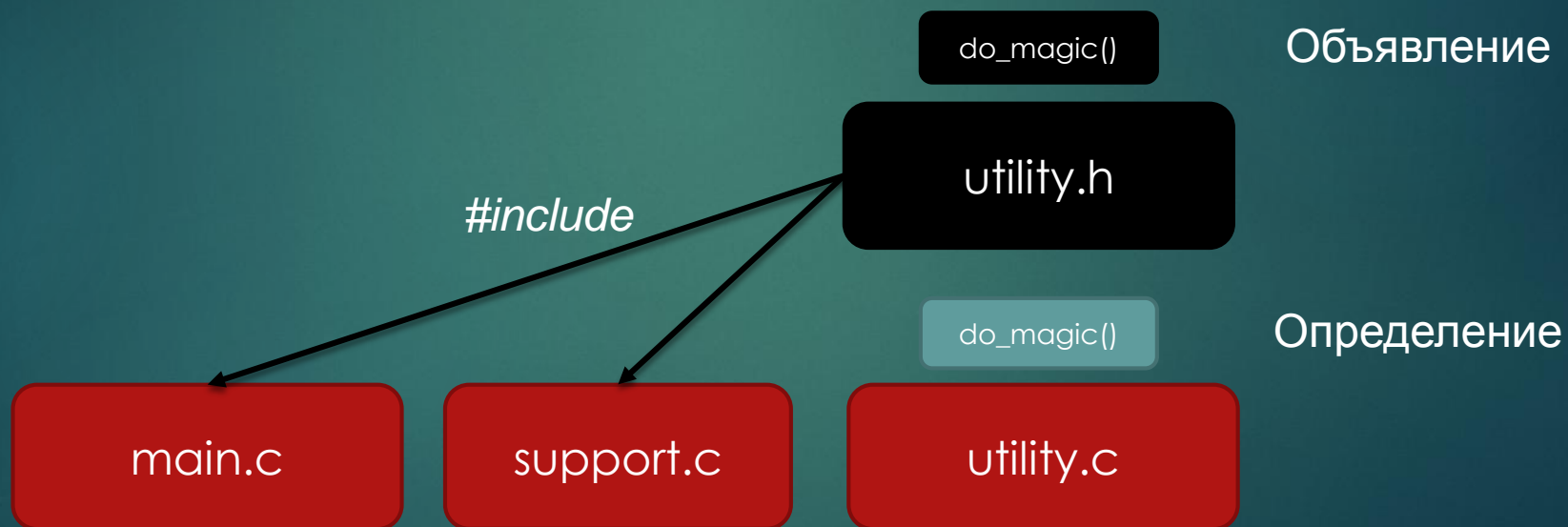
```
a.c x main.c x
1 #include "stdio.h"
2
3 void do_something(int arg)
4 {
5     ...printf("do_something with arg %d\n", arg);
6 }
```

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл Правка Вид Терминал Вкладки Справка
[snork@snork-pc gcc-test]$ gcc -c main.c -o main.o
[snork@snork-pc gcc-test]$ gcc -c a.c -o a.o
[snork@snork-pc gcc-test]$ gcc main.o a.o -o main
[snork@snork-pc gcc-test]$ ./main
do_something with arg 1
[snork@snork-pc gcc-test]$
```

Использование хедеров

Если функция используется во многих файлах, то её объявление придется писать в каждом из них.

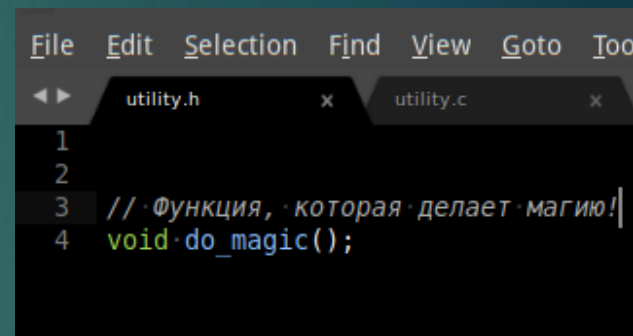
Если в описании функции нужно что-то изменить, то придется менять все объявления.



Использование хедеров

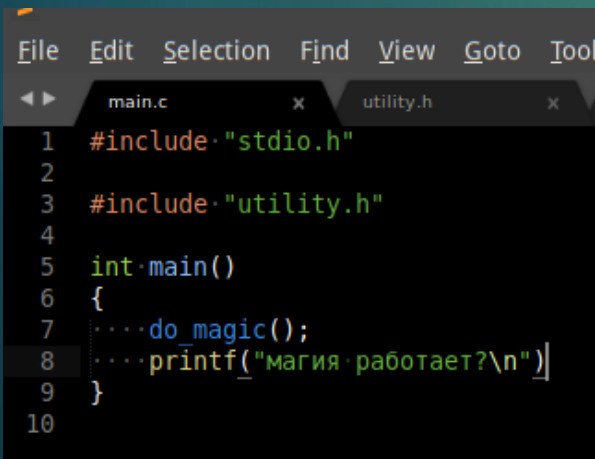
Если функция используется во многих файлах, то её объявление придется писать в КАЖДОМ из них.

Если в описании функции нужно что-то изменить, то придется менять все объявления.



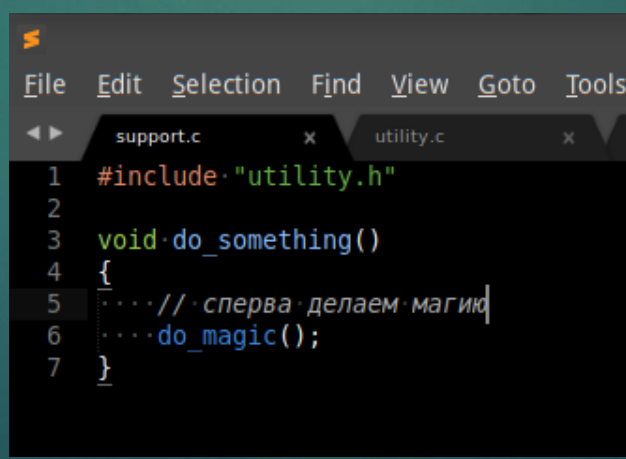
```
File Edit Selection Find View Goto Tools
utility.h x utility.c x
1
2
3 // Функция, которая делает магию!
4 void do_magic();
```

utility.h



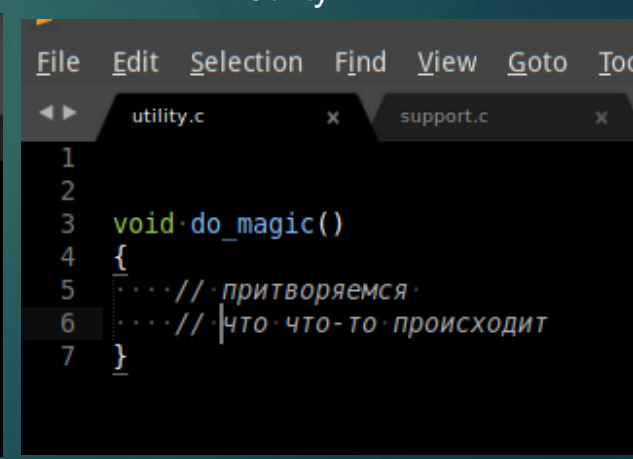
```
File Edit Selection Find View Goto Tools
main.c x utility.h x
1 #include "stdio.h"
2
3 #include "utility.h"
4
5 int main()
6 {
7     do_magic();
8     printf("магия работает?\n");
9 }
10
```

main.c



```
File Edit Selection Find View Goto Tools
support.c x utility.c x
1 #include "utility.h"
2
3 void do_something()
4 {
5     // сперва делаем магию
6     do_magic();
7 }
```

support.c



```
File Edit Selection Find View Goto Tools
utility.c x support.c x
1
2
3 void do_magic()
4 {
5     // притворяемся
6     // что-то происходит
7 }
```

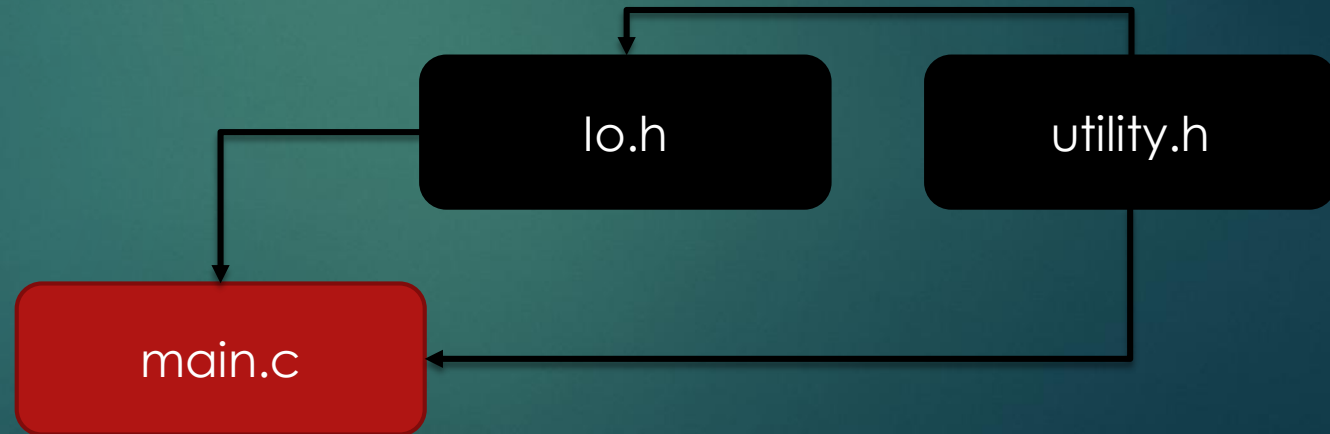
utility.c

Include guard

Если хедеров в проекте много, и они включают друг друга, очень легко запутаться и включить один и тот же заголовок несколько раз.

Если в заголовках используются только объявления – это не страшно, так как их может быть сколько угодно в программе

Но если в заголовке определяется например структура, то определение структуры должно быть одно в каждой **единице компиляции**



Include guard

utility.h

io.h

main.c

```
io.h
1 #include "utility.h"
2
3 void some_io_function();
```

```
utility.h
1
2
3 typedef struct
4 {
5     int a;
6     int b;
7     int c;
8 } my_struct_t;
```

```
main.c
1 #include "io.h"
2 #include "utility.h"
3
4 int main()
5 {
6
7
8     return 0;
9 }
```

```
[snork@snork-pc gcc-test]$ gcc main.c -o main
In file included from main.c:2:0:
utility.h:8:3: ошибка: несовместимые типы для «my_struct_t»
    } my_struct_t;
      ^~~~~~
In file included from io.h:1:0,
                  from main.c:1:
utility.h:8:3: замечание: здесь была предыдущая декларация «my_struct_t»
    } my_struct_t;
      ^~~~~~
[snork@snork-pc gcc-test]$
```

Include guard

utility.h

io.h

main.c

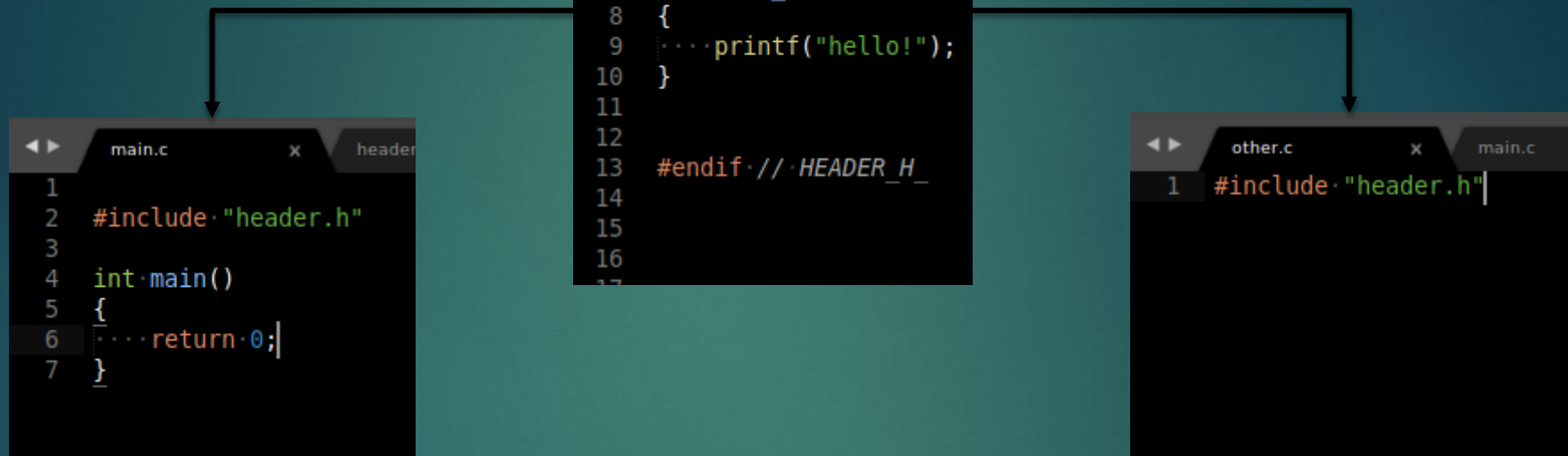
```
main.c x io.h
1 #include "io.h"
2 #include "utility.h"
3
4 int main()
5 {
6
7
8     return 0;
9 }
```

```
main.c x io.h
1 #ifndef UTILITY_H_
2 #define UTILITY_H_
3
4
5 #include "utility.h"
6
7 void some_io_function();
8
9
10 #endif // UTILITY_H_
```

```
main.c x io.h
1 #ifndef UTILITY_H_
2 #define UTILITY_H_
3
4
5 typedef struct
6 {
7     int a;
8     int b;
9     int c;
10 } my_struct_t;
11
12
13 #endif // UTILITY_H_
```

```
Терминал - snork@snork-pc: ~/Desktop/
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ gcc main.c -o main
[snork@snork-pc gcc-test]$
```

Inline



```
Терминал - snork@snork-pc: ~/Desktop/gcc-test  
Файл  Правка  Вид  Терминал  Вкладки  Справка  
[snork@snork-pc gcc-test]$ gcc -c main.c -o main.o  
[snork@snork-pc gcc-test]$ gcc -c other.c -o other.o  
[snork@snork-pc gcc-test]$ gcc main.o other.o -o the_program  
other.o: In function `the_function':  
other.c:(.text+0x0): multiple definition of `the_function'  
main.o:main.c:(.text+0x0): first defined here  
collect2: ошибка: выполнение ld завершилось с кодом возврата 1  
[snork@snork-pc gcc-test]$
```


Inline

```
header.h
1  #ifndef HEADER_H_
2  #define HEADER_H_
3
4  #include "stdio.h"
5
6
7  void the_function()
8  {
9      printf("hello!");
10 }
11
12
13 #endif // HEADER_H_
14
15
16
17
```

Символы main.o

Символы other.o

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ objdump -t main.o

main.o:      формат файла elf64-x86-64

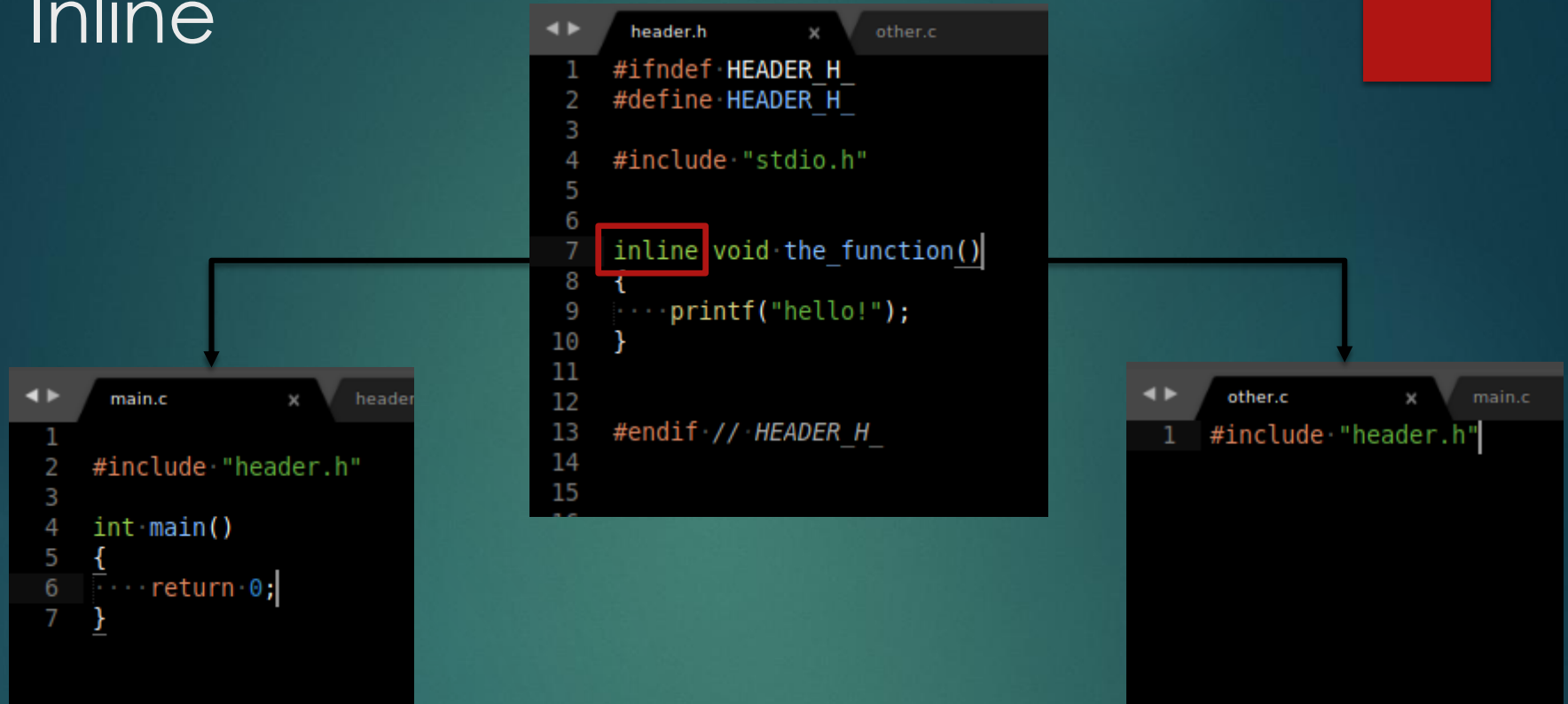
SYMBOL TABLE:
0000000000000000 l   df *ABS* 0000000000000000 main.c
0000000000000000 l   d  .text 0000000000000000 .text
0000000000000000 l   d  .data 0000000000000000 .data
0000000000000000 l   d  .bss 0000000000000000 .bss
0000000000000000 l   d  .rodata 0000000000000000 .rodata
0000000000000000 l   d  .note.GNU-stack 0000000000000000 .note.GNU-stack
0000000000000000 l   d  .eh_frame 0000000000000000 .eh_frame
0000000000000000 l   d  .comment 0000000000000000 .comment
0000000000000000 g   F  .text 0000000000000018 the_function
0000000000000000 *UND* 0000000000000000 _GLOBAL_OFFSET_TABLE_
0000000000000000 *UND* 0000000000000000 printf
0000000000000018 g   F  .text 000000000000000b main
```

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ objdump -t other.o

other.o:      формат файла elf64-x86-64

SYMBOL TABLE:
0000000000000000 l   df *ABS* 0000000000000000 other.c
0000000000000000 l   d  .text 0000000000000000 .text
0000000000000000 l   d  .data 0000000000000000 .data
0000000000000000 l   d  .bss 0000000000000000 .bss
0000000000000000 l   d  .rodata 0000000000000000 .rodata
0000000000000000 l   d  .note.GNU-stack 0000000000000000 .note.GNU-stack
0000000000000000 l   d  .eh_frame 0000000000000000 .eh_frame
0000000000000000 l   d  .comment 0000000000000000 .comment
0000000000000000 g   F  .text 0000000000000018 the_function
0000000000000000 *UND* 0000000000000000 _GLOBAL_OFFSET_TABLE_
0000000000000000 *UND* 0000000000000000 printf
```

Inline



```
Терминал - snork@snork-pc: ~/Desktop/gcc-test  
Файл  Правка  Вид  Терминал  Вкладки  Справка  
[snork@snork-pc gcc-test]$ gcc -c main.c -o main.o  
[snork@snork-pc gcc-test]$ gcc -c other.c -o other.o  
[snork@snork-pc gcc-test]$ gcc main.o other.o -o the_program  
[snork@snork-pc gcc-test]$
```

Inline

Символы main.o

```
header.h  x  other.c
1  #ifndef HEADER_H_
2  #define HEADER_H_
3
4  #include "stdio.h"
5
6
7  inline void the_function()
8  {
9      ...printf("hello!");
10 }
11
12
13 #endif // HEADER_H_
14
```

```
[snork@snork-pc gcc-test]$ objdump -t main.o
```

```
main.o:      формат файла elf64-x86-64
```

```
SYMBOL TABLE:
```

0000000000000000	1	df	*ABS*	0000000000000000	main.c
0000000000000000	1	d	.text	0000000000000000	.text
0000000000000000	1	d	.data	0000000000000000	.data
0000000000000000	1	d	.bss	0000000000000000	.bss
0000000000000000	1	d	.note.GNU-stack	0000000000000000	.note.GNU-stack
0000000000000000	1	d	.eh_frame	0000000000000000	.eh_frame
0000000000000000	1	d	.comment	0000000000000000	.comment
0000000000000000	g	F	.text	000000000000000b	main

```
[snork@snork-pc gcc-test]$
```

Символы other.o

Файл Правка Вид Терминал Вкладки Справка

```
[snork@snork-pc gcc-test]$ objdump -t other.o
```

```
other.o:      формат файла elf64-x86-64
```

```
SYMBOL TABLE:
```

0000000000000000	1	df	*ABS*	0000000000000000	other.c
0000000000000000	1	d	.text	0000000000000000	.text
0000000000000000	1	d	.data	0000000000000000	.data
0000000000000000	1	d	.bss	0000000000000000	.bss
0000000000000000	1	d	.note.GNU-stack	0000000000000000	.note.GNU-stack
0000000000000000	1	d	.comment	0000000000000000	.comment

```
[snork@snork-pc gcc-test]$
```

static

main.c

```
other.c x main.c x
1
2 void write()
3 {
4
5 }
6
7
8
9 int main()
10 {
11     ...return 0;
12 }
```

other.c

```
other.c x ma
1
2
3 void write()
4 {
5     ...
6 }
```

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ gcc -c other.c -o other.o
[snork@snork-pc gcc-test]$ gcc -c main.c -o main.o
[snork@snork-pc gcc-test]$ gcc main.o other.o -o the_program
other.o: In function `write':
other.c:(.text+0x0): multiple definition of `write'
main.o:main.c:(.text+0x0): first defined here
collect2: ошибка: выполнение ld завершилось с кодом возврата 1
[snork@snork-pc gcc-test]$
```

static

main.c

```
other.c x main.c x
1
2 void write()
3 {
4
5 }
6
7
8
9 int main()
10 {
11     ...return 0;
12 }
```

other.c

```
other.c x ma
1
2
3 void write()
4 {
5     ...
6 }
```

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ gcc -c other.c -o other.o
[snork@snork-pc gcc-test]$ gcc -c main.c -o main.o
[snork@snork-pc gcc-test]$ gcc main.o other.o -o the_program
other.o: In function `write':
other.c:(.text+0x0): multiple definition of `write'
main.o:main.c:(.text+0x0): first defined here
collect2: ошибка: выполнение ld завершилось с кодом возврата 1
[snork@snork-pc gcc-test]$
```

static

main.c

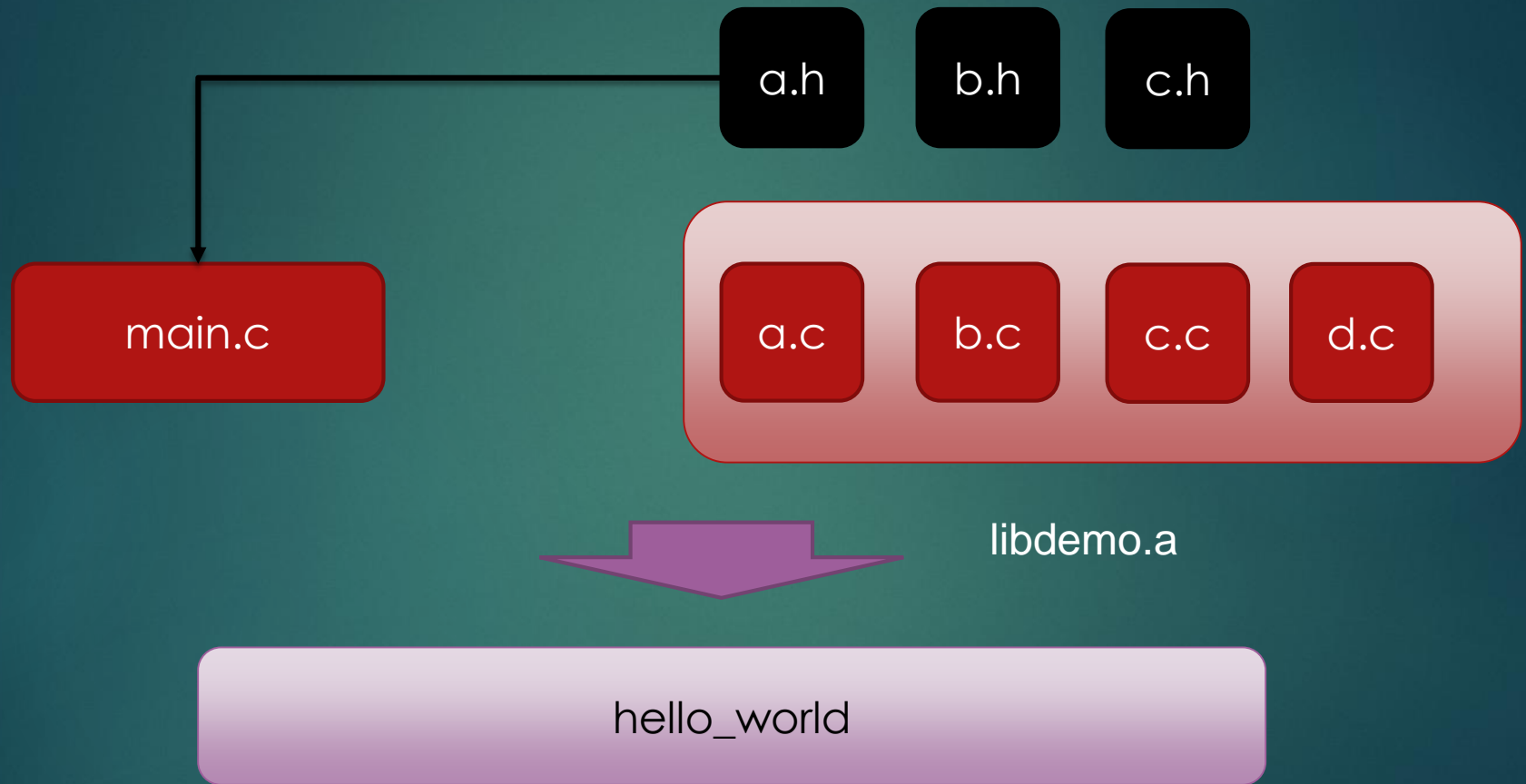
```
other.c x main.c x
1
2 void write()
3 {
4
5 }
6
7
8
9 int main()
10 {
11     ...return 0;
12 }
```

other.c

```
other.c x main.c
1
2
3 static void write()
4 {
5
6 }
7
8
9 void use_write()
10 {
11     ...write();
12 }
```

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ gcc -c main.c -o main.o
[snork@snork-pc gcc-test]$ gcc -c other.c -o other.o
[snork@snork-pc gcc-test]$ gcc main.o other.o -o the_program
[snork@snork-pc gcc-test]$
```

Статические библиотеки



Состав программ GCC тулчейна

```
arm-raspbian-linux-gnueabi-hf-addr2line
arm-raspbian-linux-gnueabi-hf-ar
arm-raspbian-linux-gnueabi-hf-as
arm-raspbian-linux-gnueabi-hf-c++
arm-raspbian-linux-gnueabi-hf-cc -> arm-raspbian-linux-gnueabi-hf-gcc
arm-raspbian-linux-gnueabi-hf-c++filt
arm-raspbian-linux-gnueabi-hf-cpp
arm-raspbian-linux-gnueabi-hf-ct-ng.config
arm-raspbian-linux-gnueabi-hf-elfedit
arm-raspbian-linux-gnueabi-hf-g++
arm-raspbian-linux-gnueabi-hf-gcc
arm-raspbian-linux-gnueabi-hf-gcc-6.4.0
arm-raspbian-linux-gnueabi-hf-gcc-ar
arm-raspbian-linux-gnueabi-hf-gcc-nm
arm-raspbian-linux-gnueabi-hf-gcc-ranlib
arm-raspbian-linux-gnueabi-hf-gcov
arm-raspbian-linux-gnueabi-hf-gcov-dump
arm-raspbian-linux-gnueabi-hf-gcov-tool
arm-raspbian-linux-gnueabi-hf-gdb
arm-raspbian-linux-gnueabi-hf-gprof
arm-raspbian-linux-gnueabi-hf-ld
arm-raspbian-linux-gnueabi-hf-ld.bfd
arm-raspbian-linux-gnueabi-hf-ldd
arm-raspbian-linux-gnueabi-hf-nm
arm-raspbian-linux-gnueabi-hf-objcopy
arm-raspbian-linux-gnueabi-hf-objdump
arm-raspbian-linux-gnueabi-hf-populate
arm-raspbian-linux-gnueabi-hf-ranlib
arm-raspbian-linux-gnueabi-hf-readelf
arm-raspbian-linux-gnueabi-hf-size
arm-raspbian-linux-gnueabi-hf-strings
arm-raspbian-linux-gnueabi-hf-strip
```

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ size main
text    data    bss     dec     hex filename
1335    520      8    1863     747 main
[snork@snork-pc gcc-test]$
```

```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ cpp main.c | grep "^[^#;]"
void some_io_function();
int main()
{
    return 0;
}
[snork@snork-pc gcc-test]$
```

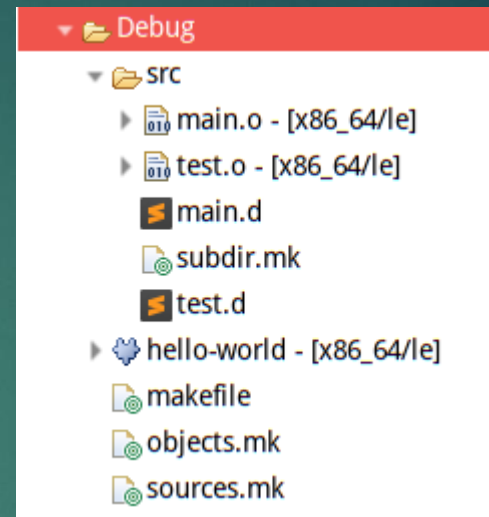
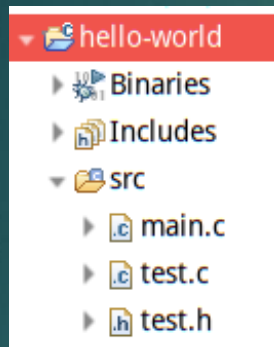
```
Терминал - snork@snork-pc: ~/Desktop/gcc-test
Файл  Правка  Вид  Терминал  Вкладки  Справка
[snork@snork-pc gcc-test]$ objdump -d main.o

main.o:      формат файла elf64-x86-64

Дизассемблирование раздела .text:

0000000000000000 <main>:
0:  55                      push    %rbp
1:  48 89 e5                mov     %rsp,%rbp
4:  b8 00 00 00 00         mov     $0x0,%eax
9:  5d                      pop     %rbp
a:  c3                      retq
[snork@snork-pc gcc-test]$
```


Что делает ЭКЛИПС?



Problems Tasks Console Properties Call Graph Search

CDT Build Console [hello-world]

02:31:13 **** Build of configuration Debug for project hello-world ****

make all

Building file: ../src/main.c

Invoking: GCC C Compiler

gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/main.d" -MT"src/main.o" -o "src/main.o" "../src/main.c"

Finished building: ../src/main.c

Building file: ../src/test.c

Invoking: GCC C Compiler

gcc -O0 -g3 -Wall -c -fmessage-length=0 -MMD -MP -MF"src/test.d" -MT"src/test.o" -o "src/test.o" "../src/test.c"

Finished building: ../src/test.c

Building target: hello-world

Invoking: GCC C Linker

gcc -o "hello-world" ./src/main.o ./src/test.o

Finished building target: hello-world