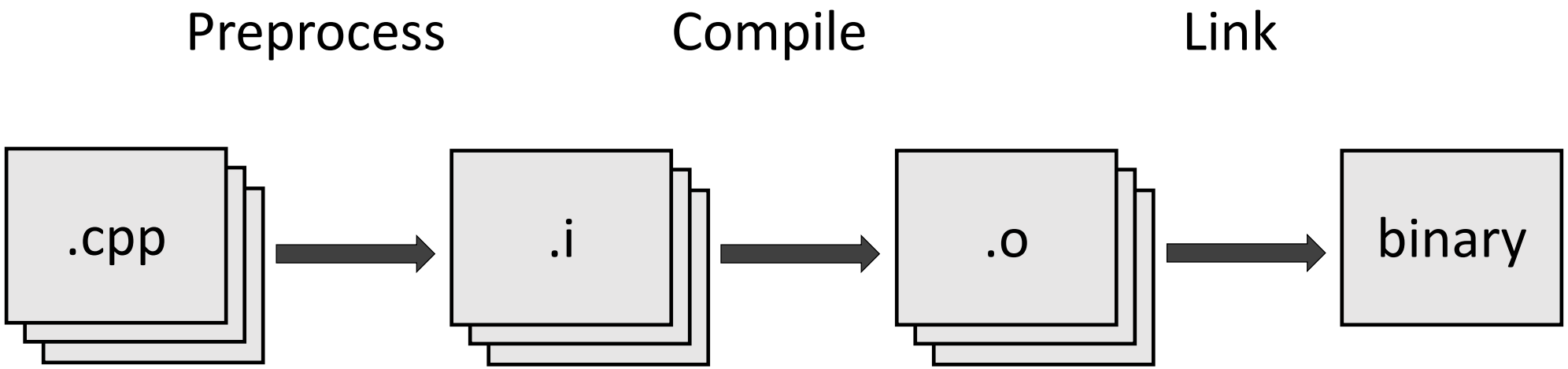
Межмодульный анализ C++ проектов

PVS-Studio

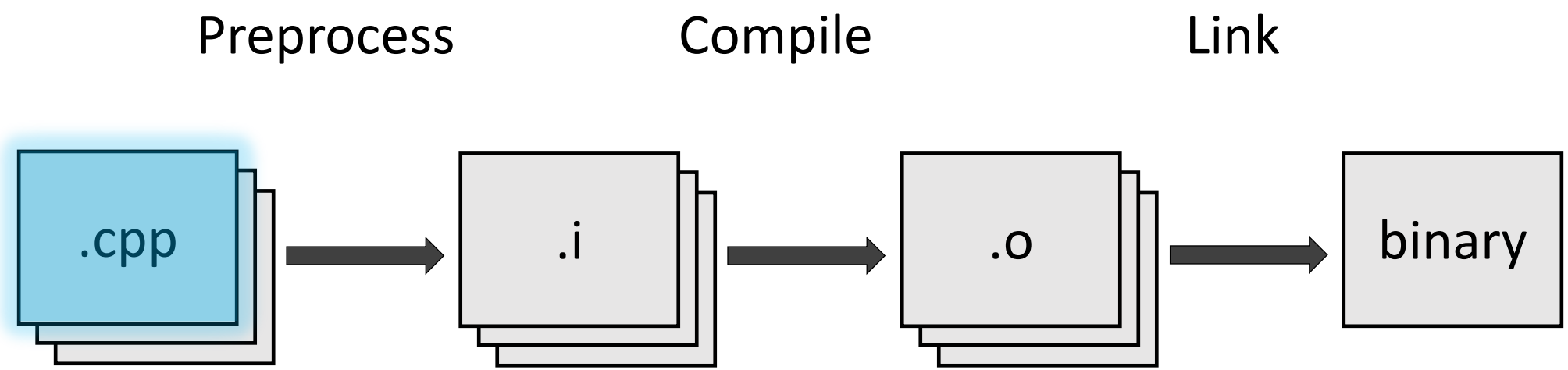
C++ Developer
PVS-Studio

Лысый Олег

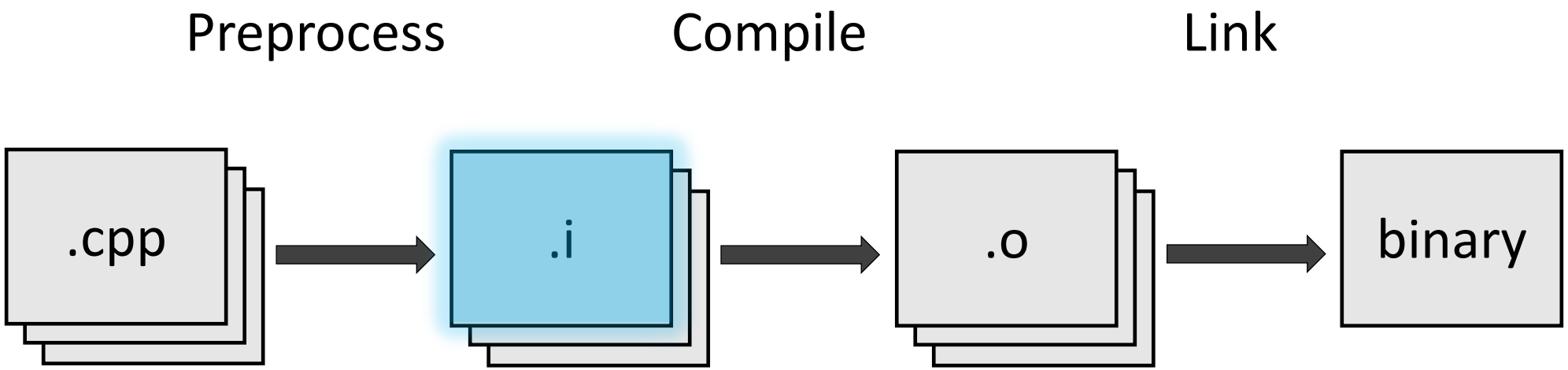
Конвейер компиляции



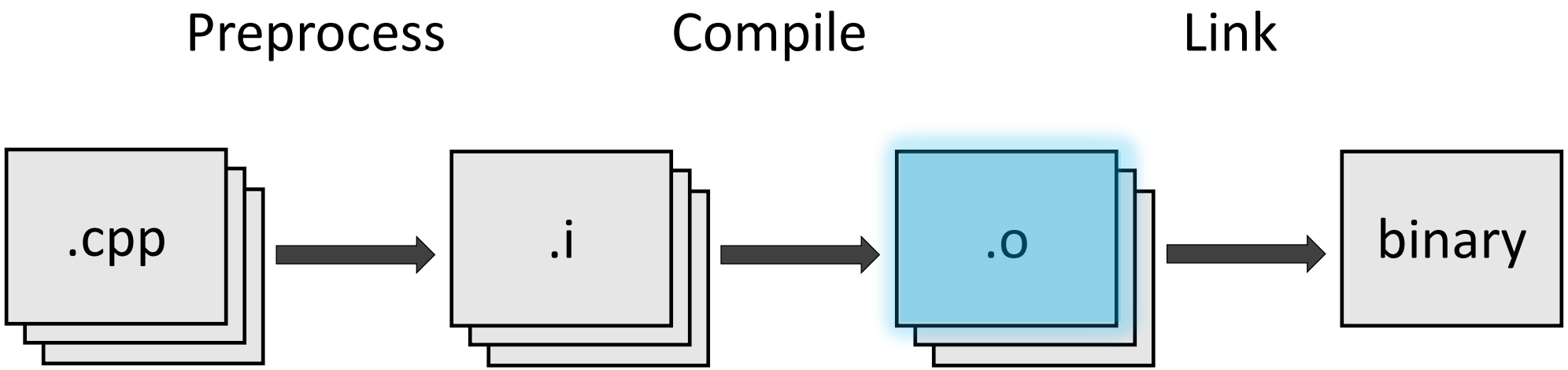
Конвейер компиляции



Конвейер компиляции



Конвейер компиляции



Символы

```
struct Cat
{
  static int x;
};

Cat::x = 0;

int foo(int arg)
{
  static float symbol = 3.14f;
  static char x = 2;
  static Cat dog { };

  return 0;
}
```

sym.cpp

<x, int, static>

<foo, int(int)>

<symbol, float, static>
<x, char, static>
<dog, Cat, static>

Таблицы символов

```
% llvm-readelf -s -C sym.cpp.o
```

```
Symbol table '.symtab' contains 8 entries:
   Num:    Value          Size Type    Bind   Vis       Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT   UND
     1: 0000000000000000     0 FILE    LOCAL  DEFAULT   ABS sym.cpp
     2: 0000000000000004     1 OBJECT  LOCAL  DEFAULT     4 foo(int)::x
     3: 0000000000000000     1 OBJECT  LOCAL  DEFAULT     5 foo(int)::dog
     4: 0000000000000000     4 OBJECT  LOCAL  DEFAULT     4 foo(int)::symbol
     5: 0000000000000000     0 SECTION LOCAL  DEFAULT     2 .text
     6: 0000000000000000    22 FUNC    GLOBAL DEFAULT     2 foo(int)
     7: 0000000000000000     0 NOTYPE  GLOBAL DEFAULT   UND Cat::x
```

Linkage

no linkage
internal linkage
external linkage
module linkage

```
int foo(int x1 /* no linkage */)
{
  int x4;       // no linkage

  struct A;     // no linkage
}
```

Linkage

no linkage
internal linkage
external linkage
module linkage

```
static int x3;      // internal
const int x4 = 0;   // internal

void bar()
{
  static int x5; // internal
}

namespace // internal
{
  void internal(int a, int b)
  {
  }
}
```

Linkage

no linkage
internal linkage
external linkage
module linkage

```
extern int x2; // external
void bar();    // external
```

Linkage

no linkage
internal linkage
external linkage
module linkage

```
export module foo;

import :counter;
import export :arith;

export int get_value()
{
  return ::counter;
}
```

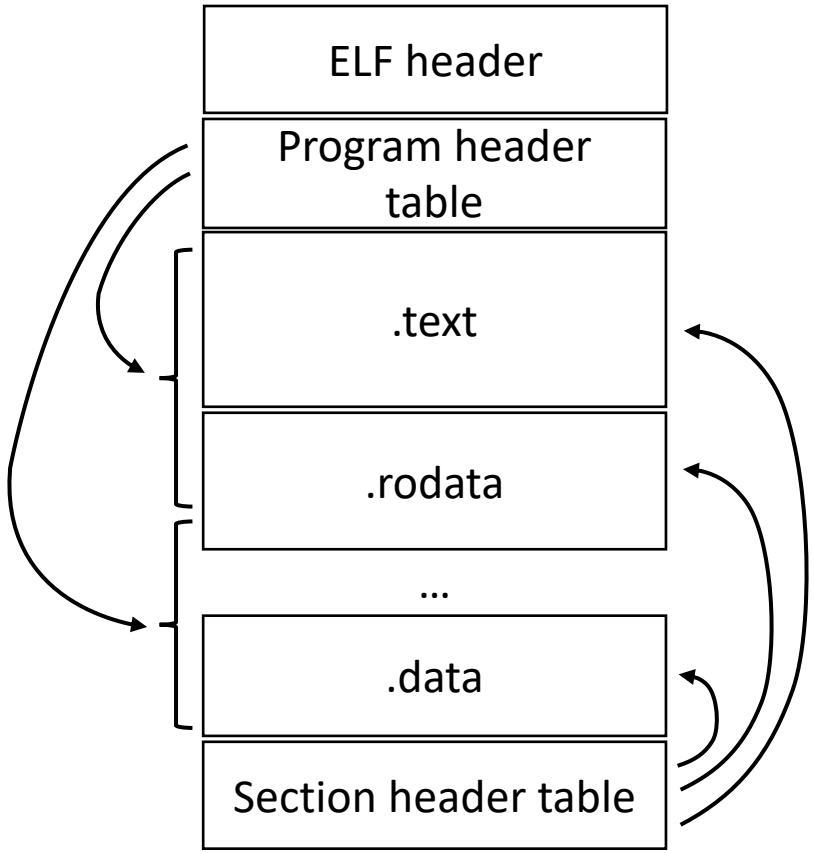
C++20

Object Files

ELF

COFF

Mach-o

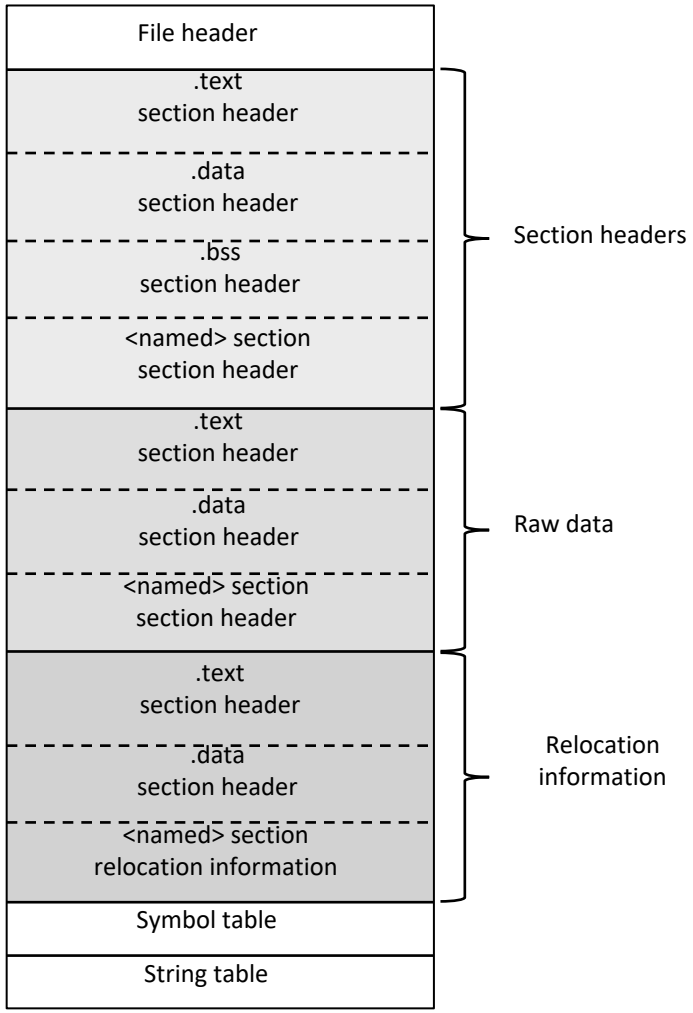


Object Files

ELF

COFF

Mach-o

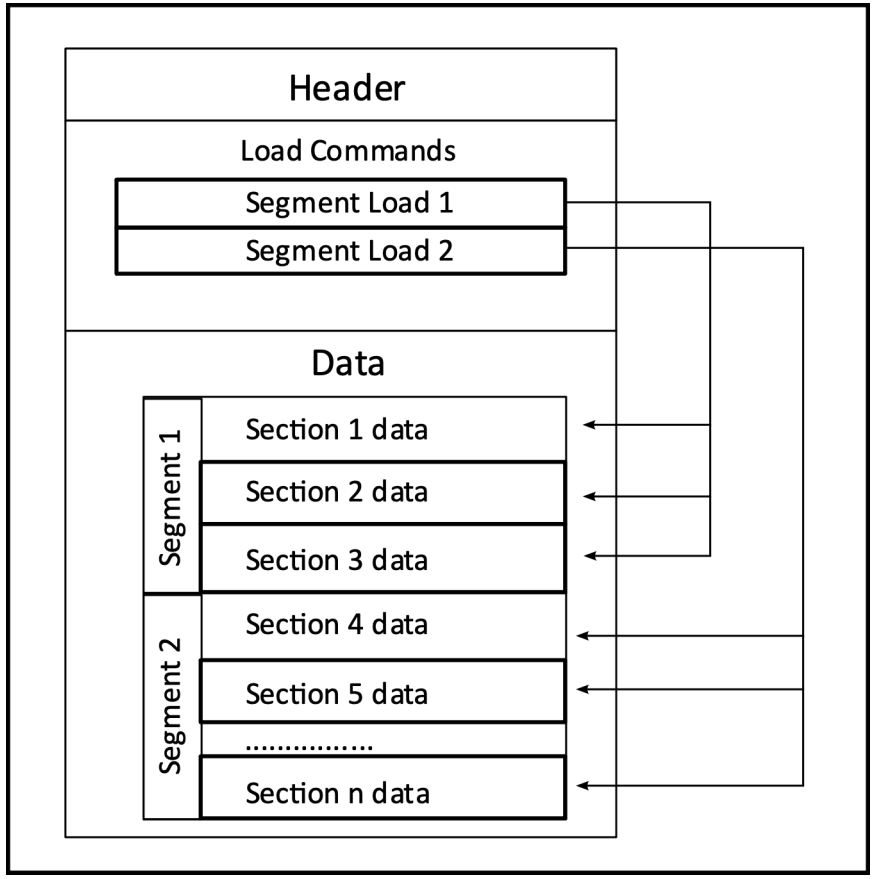


Object Files

ELF

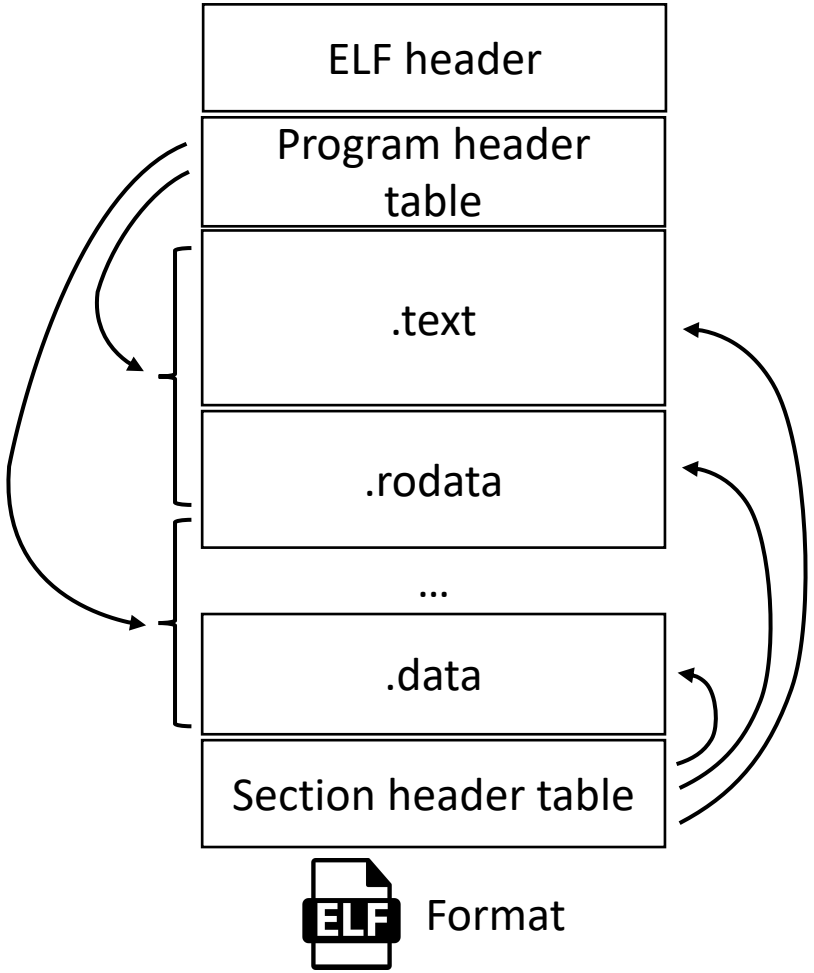
COFF

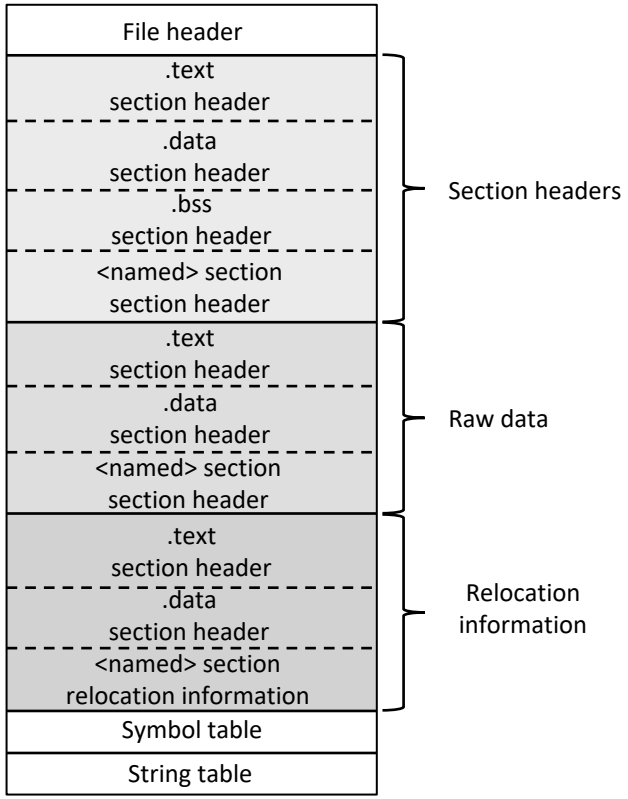
Mach-o



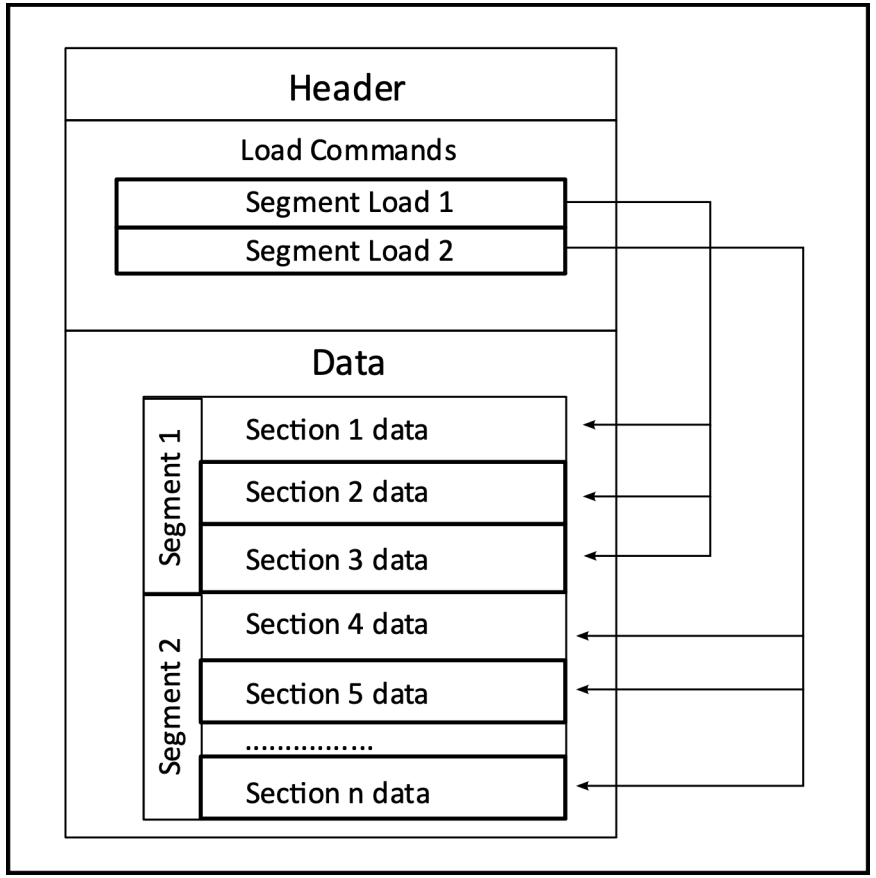
Mach-o

Object Files





COFF



Mach-o

Межмодульный анализ в компиляторах

Оптимизации времени связывания

LTO – Link Time Optimizations

Оптимизации времени связывания

- удаление мертвого кода
- разворот циклов
- раскручивание хвостовой рекурсии
- подстановка результатов вычислений на этапе компиляций
- и т. д.

Оптимизации времени связывания

A.cpp

```
void eFree(void *ptr);

int *foo();

int main()
{
  auto p = foo();
  memset(p, 0,
         sizeof(int));
  eFree(p);
}
```

B.cpp

```
void eFree(void *ptr)
{
  free(ptr);
}

int* foo()
{
  int *ret = (int*) malloc(
                      sizeof(int));
  return ret;
}
```

Оптимизации времени связывания

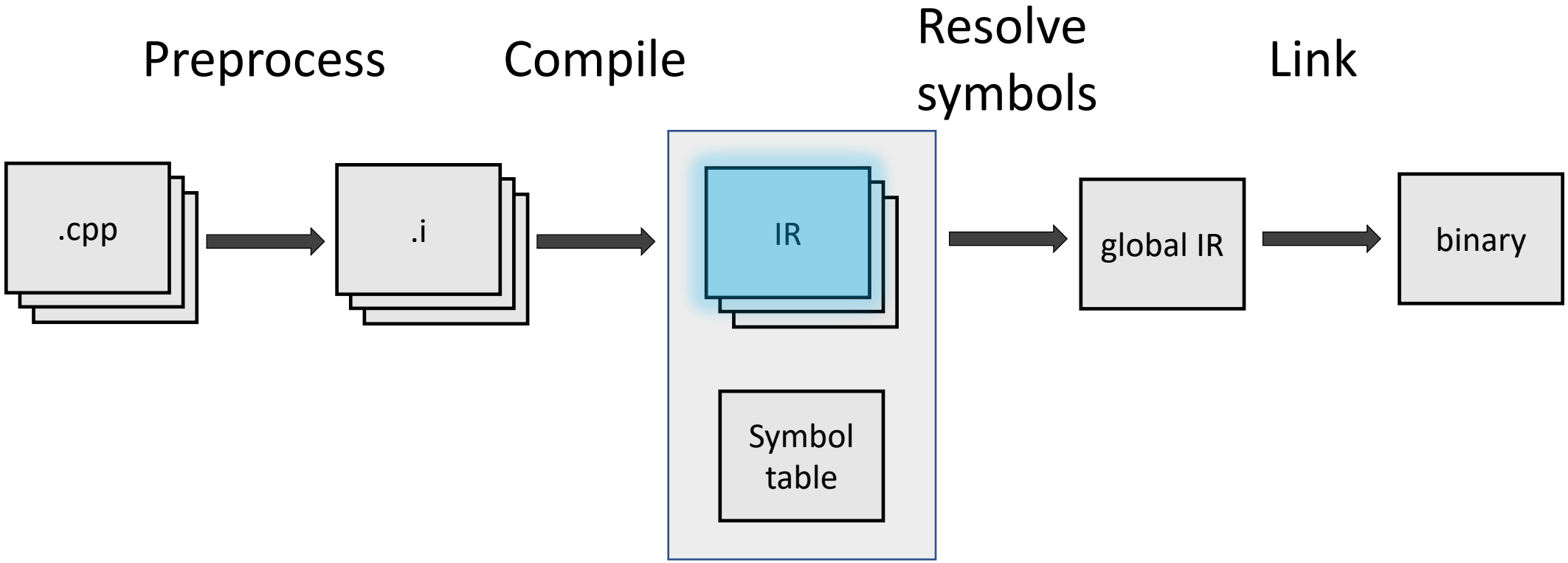
A.cpp

```
void eFree(void *ptr);

int *foo();

int main()
{
  auto p = foo();
  memset(p, 0,
         sizeof(int));
  eFree(p);
}
```

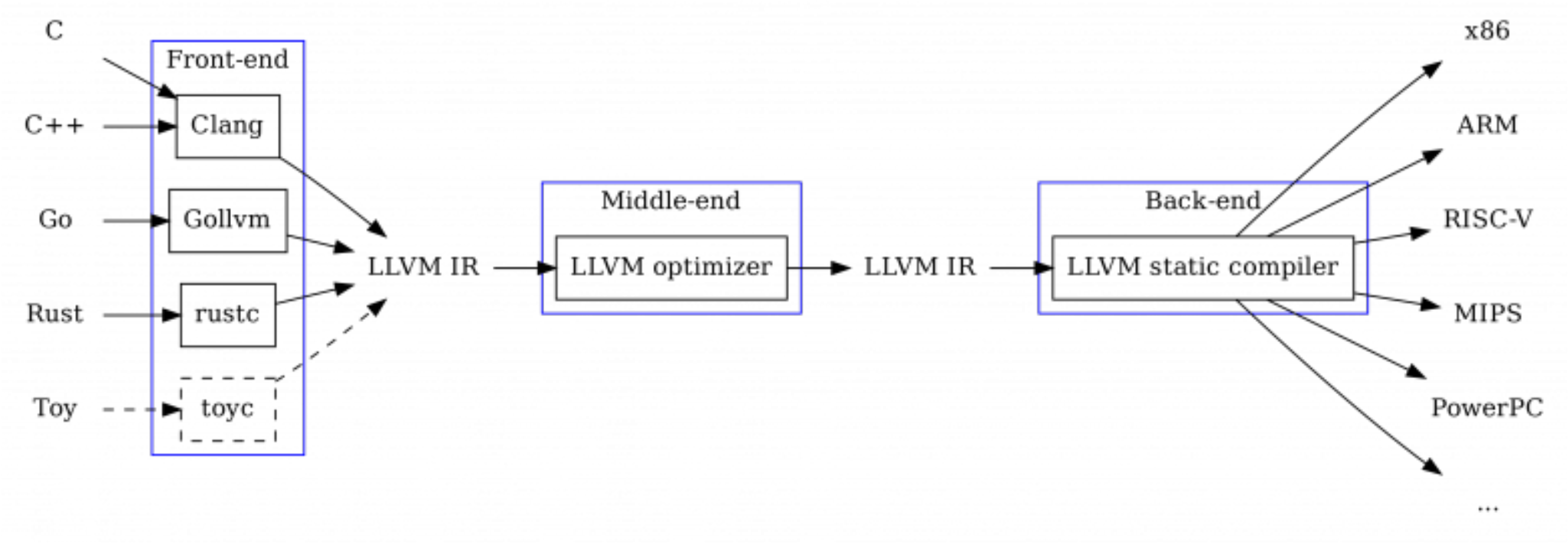
B.cpp

```
void eFree(void *ptr)
{
  free(ptr);
}

int* foo()
{
  int *ret = (int*) malloc(
                      sizeof(int));
  return ret;
}
```

LTO в Clang


Preprocess
Compile
Resolve symbols
Link
.cpp
.i
IR
Symbol table
global IR
binary

Intermediate Representation



Intermediate Representation

```
int add(int x, int y)
{
  return x + y;
}
```

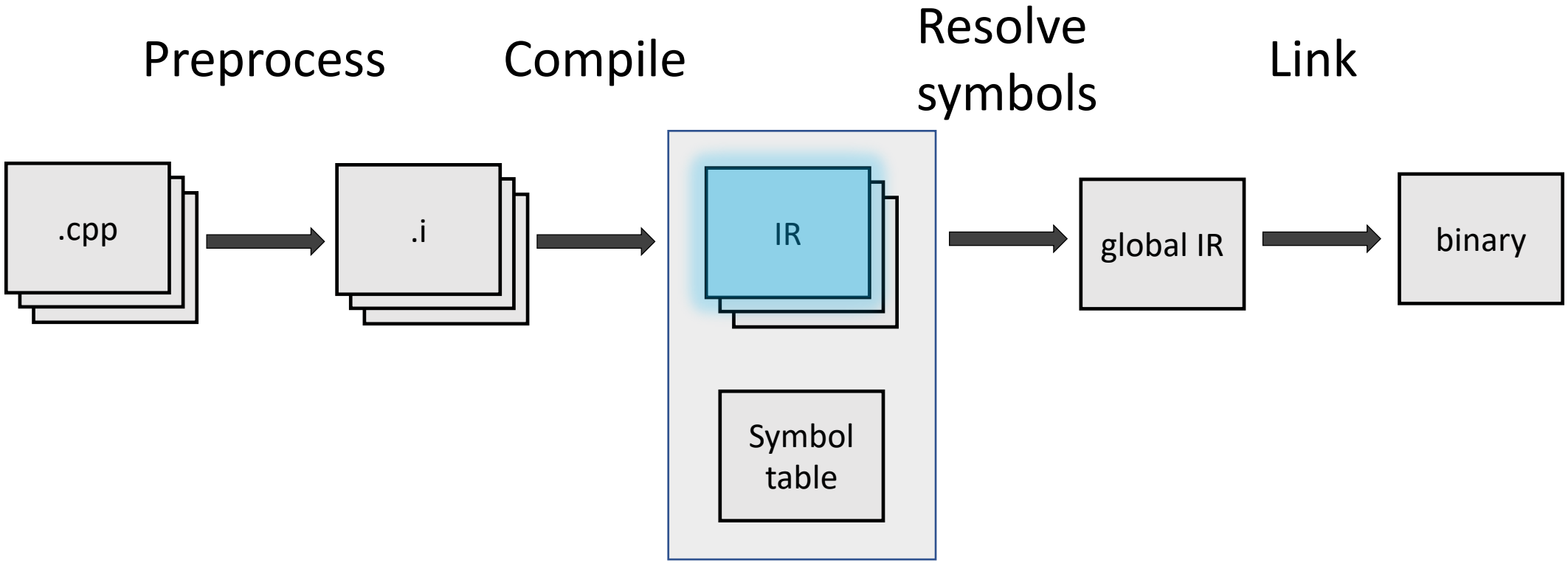
simple.cpp

```
define dso_local i32 @_Z3addii(i32 %0, i32 %1) #0 {
  %3 = add nsw i32 %0, %1
  ret i32 %3
}
```

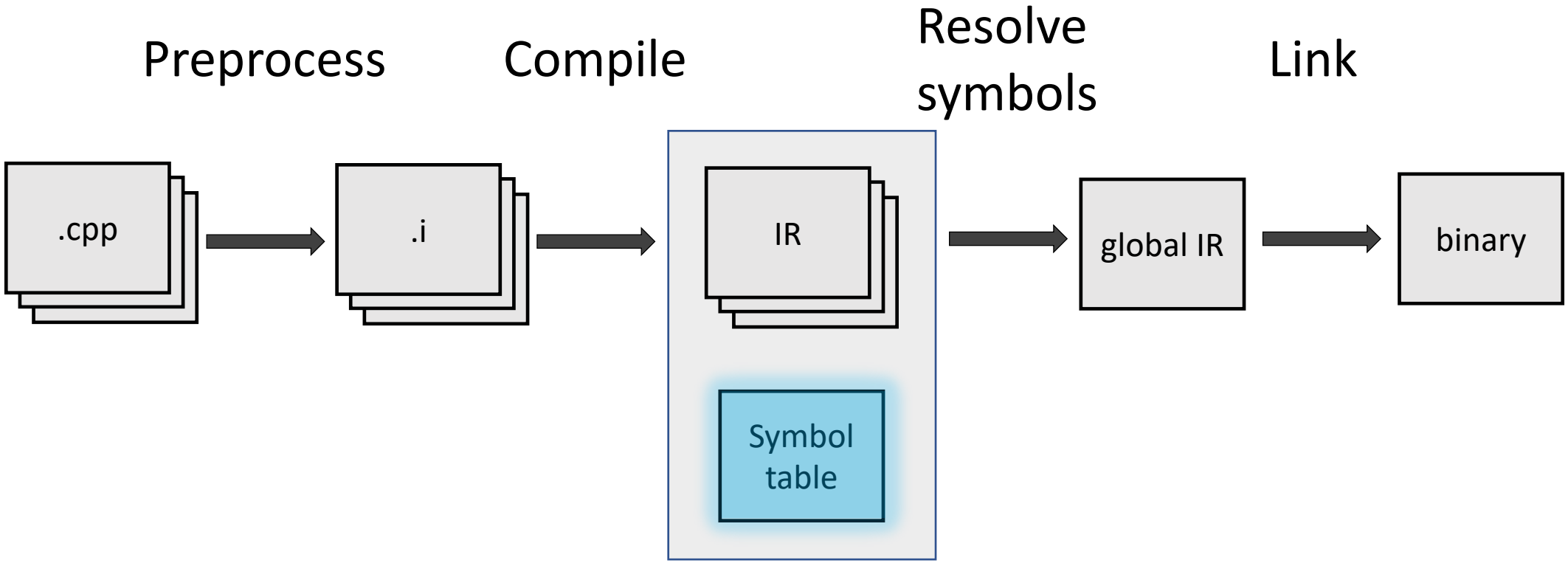
simple.cpp.ll

```
$> clang++ -S -emit-llvm -Xclang -disable-O0-optnone simple.cpp –o simple.cpp.ll
$> opt -mem2reg -S simple.cpp.ll -o simple.cpp.ll
```

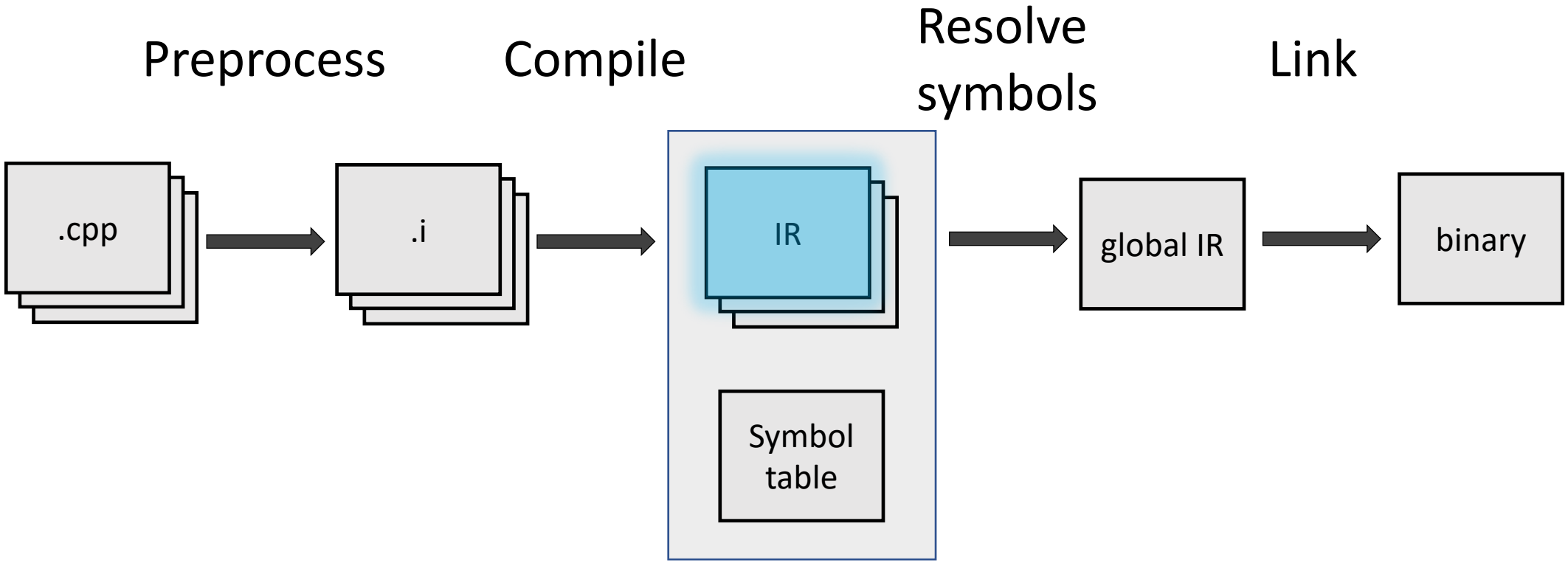
LTO в Clang


Preprocess
Compile
Resolve symbols
Link
.cpp
.i
IR
Symbol table
global IR
binary

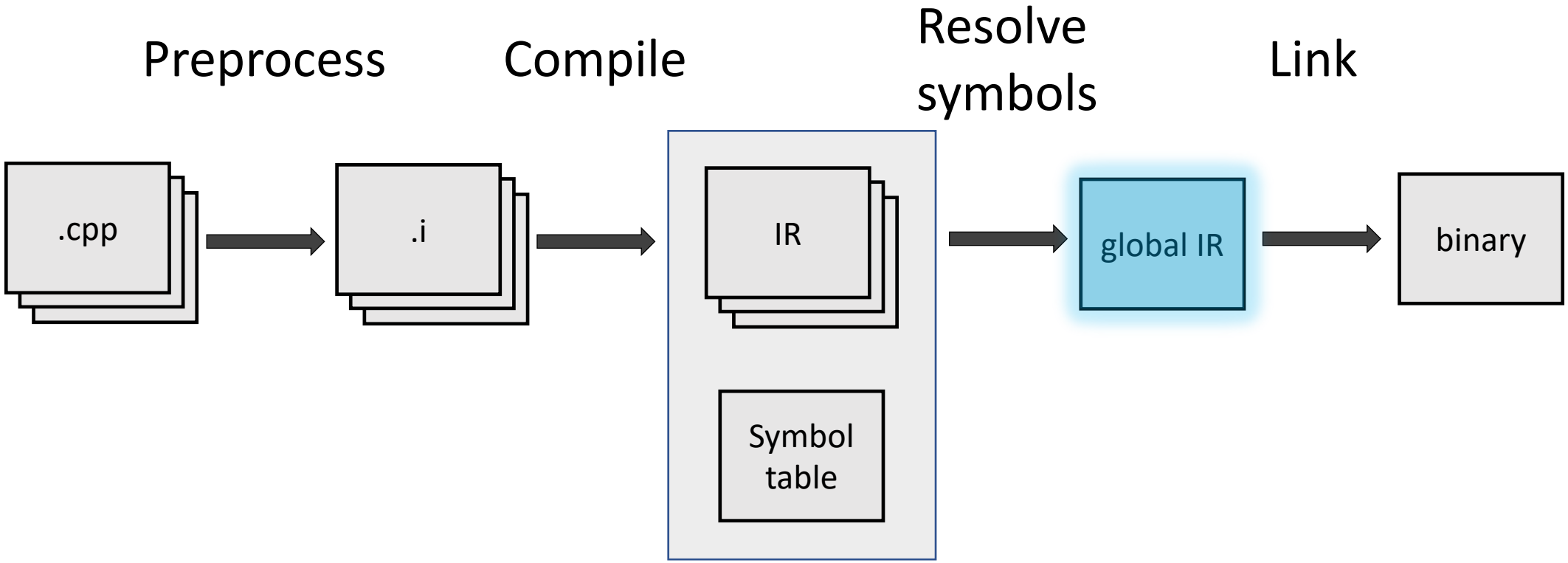
LTO в Clang


Preprocess
Compile
Resolve symbols
Link
.cpp
.i
IR
Symbol table
global IR
binary

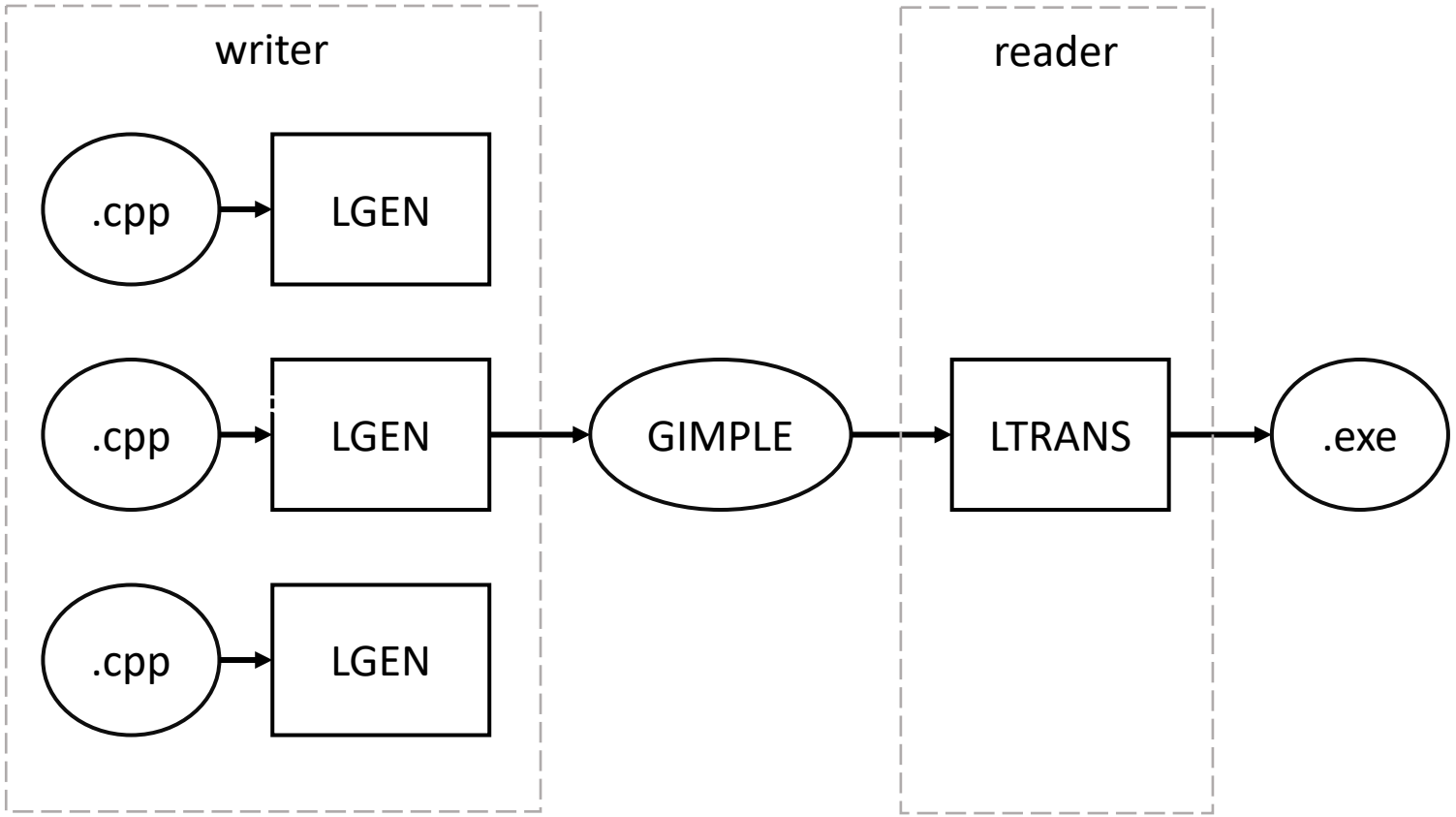
LTO в Clang


Preprocess
Compile
Resolve symbols
Link
.cpp
.i
IR
Symbol table
global IR
binary

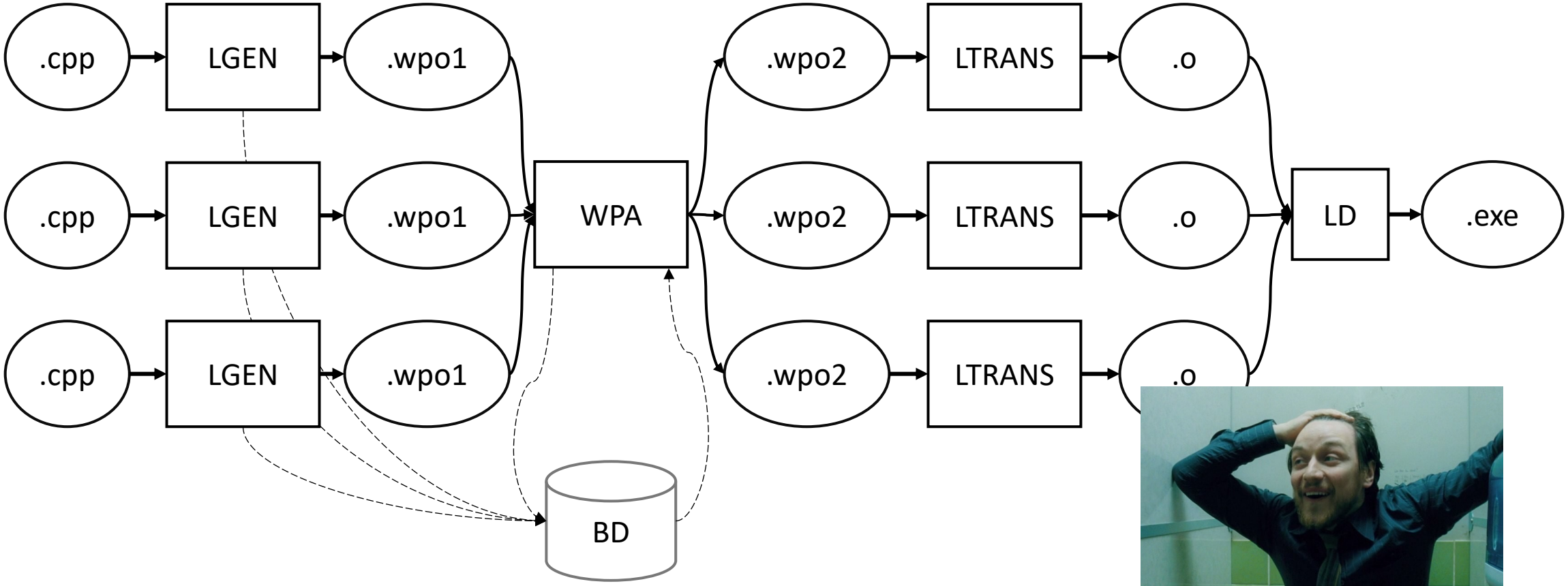
LTO в Clang


Preprocess
Compile
Resolve symbols
Link
.cpp
.i
IR
Symbol table
global IR
binary

LTO в GCC

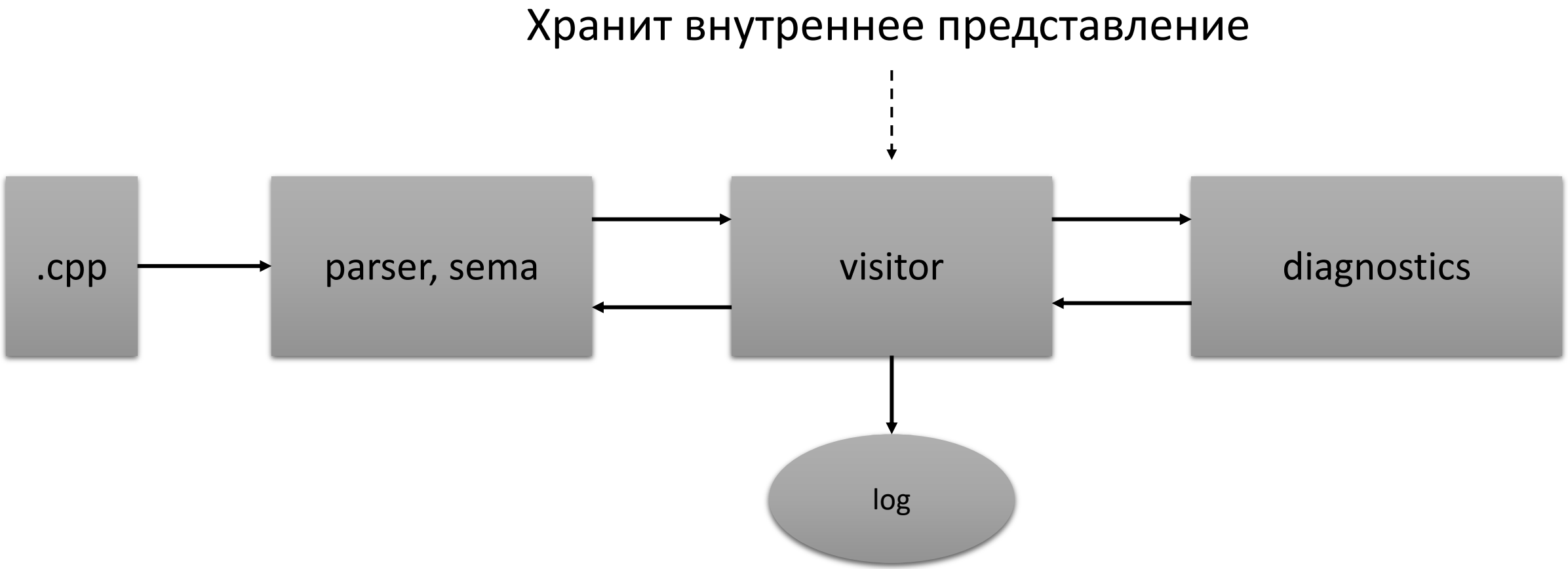


WHOPR


.cpp
LGEN
.wpo1
.cpp
LGEN
.wpo1
.cpp
LGEN
.wpo1
WPA
BD
.wpo2
LTRANS
.o
.wpo2
LTRANS
.o
.wpo2
LTRANS
.o
LD
.exe

Межмодульный анализ в статических анализаторах

Исходная архитектура

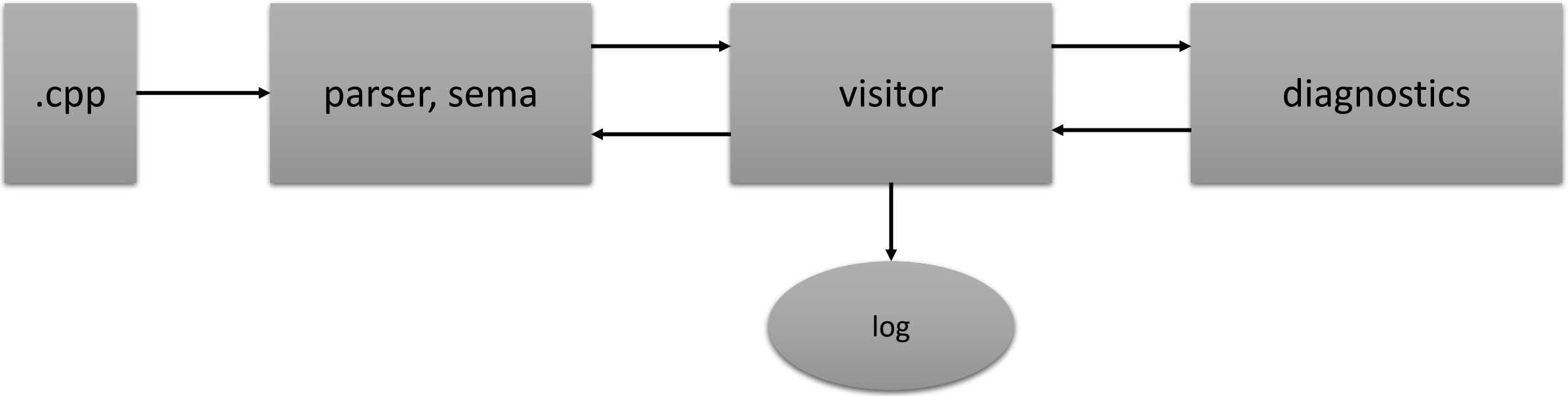


Задача: объединить данные из разных модулей

Первая проблема

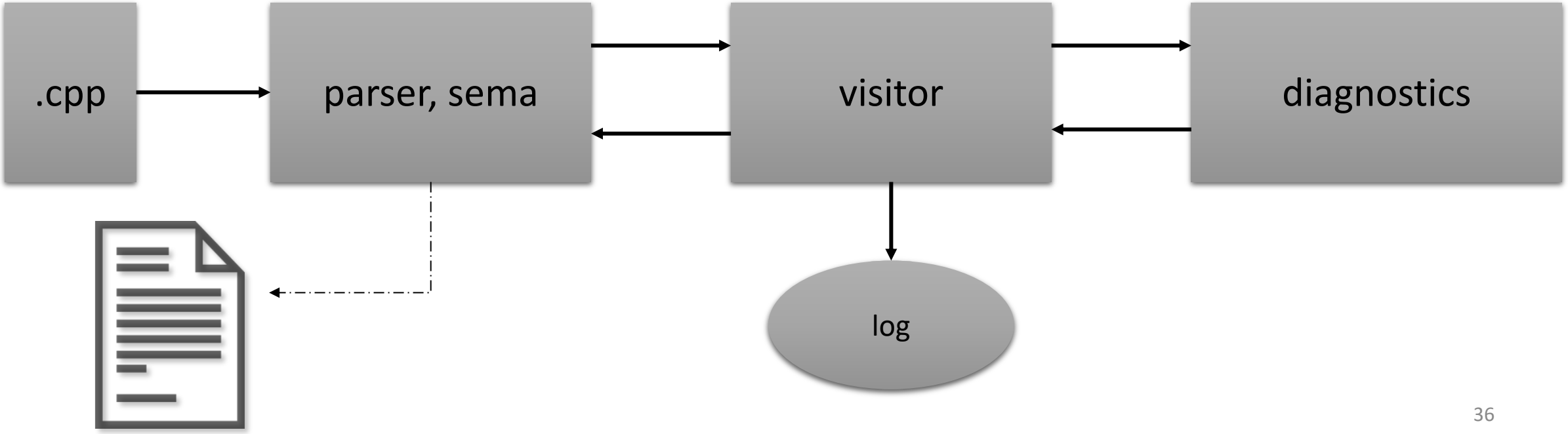
Первая проблема

Однопроходная схема анализа



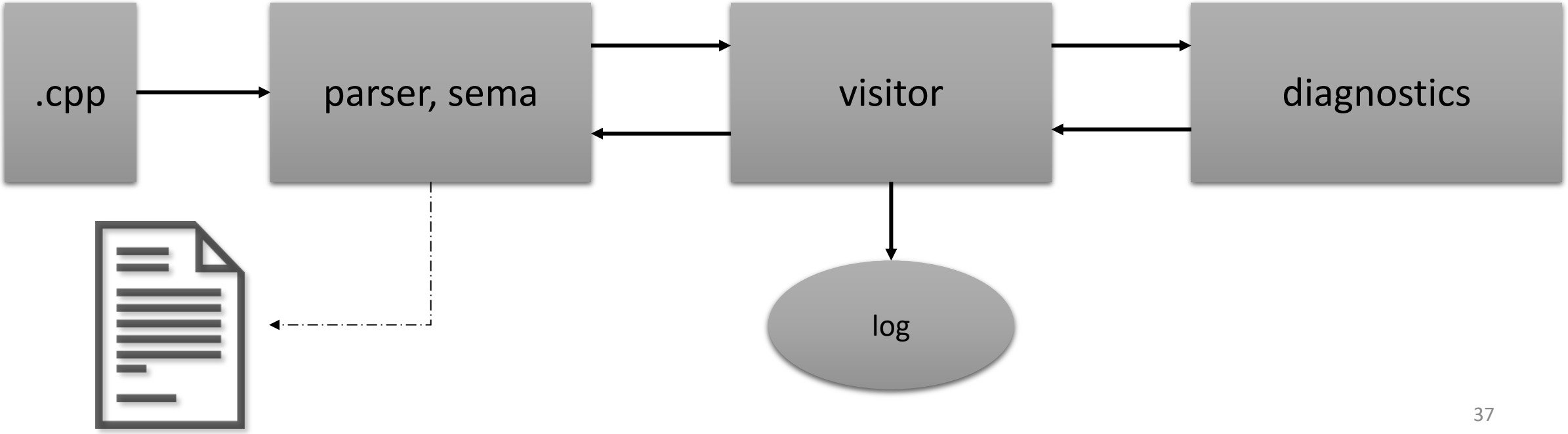
Первая проблема

Однопроходная схема анализа



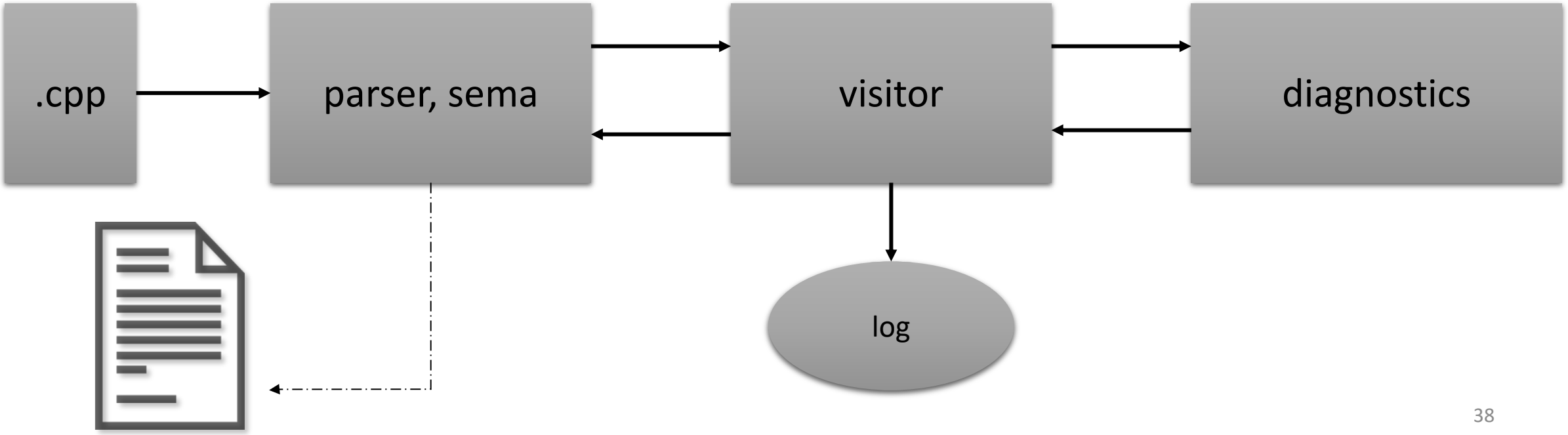
Первая проблема

Однопроходная схема анализа



Первая проблема

Однопроходная схема анализа



Трансляция

Visitor

```
struct Cat
{
  static int x;
};

Cat::x = 0;

int foo(int arg)
{
  float symbol = 3.14f;
  char x = 2;
  Cat dog { };

  return 0;
}
```

Трансляция

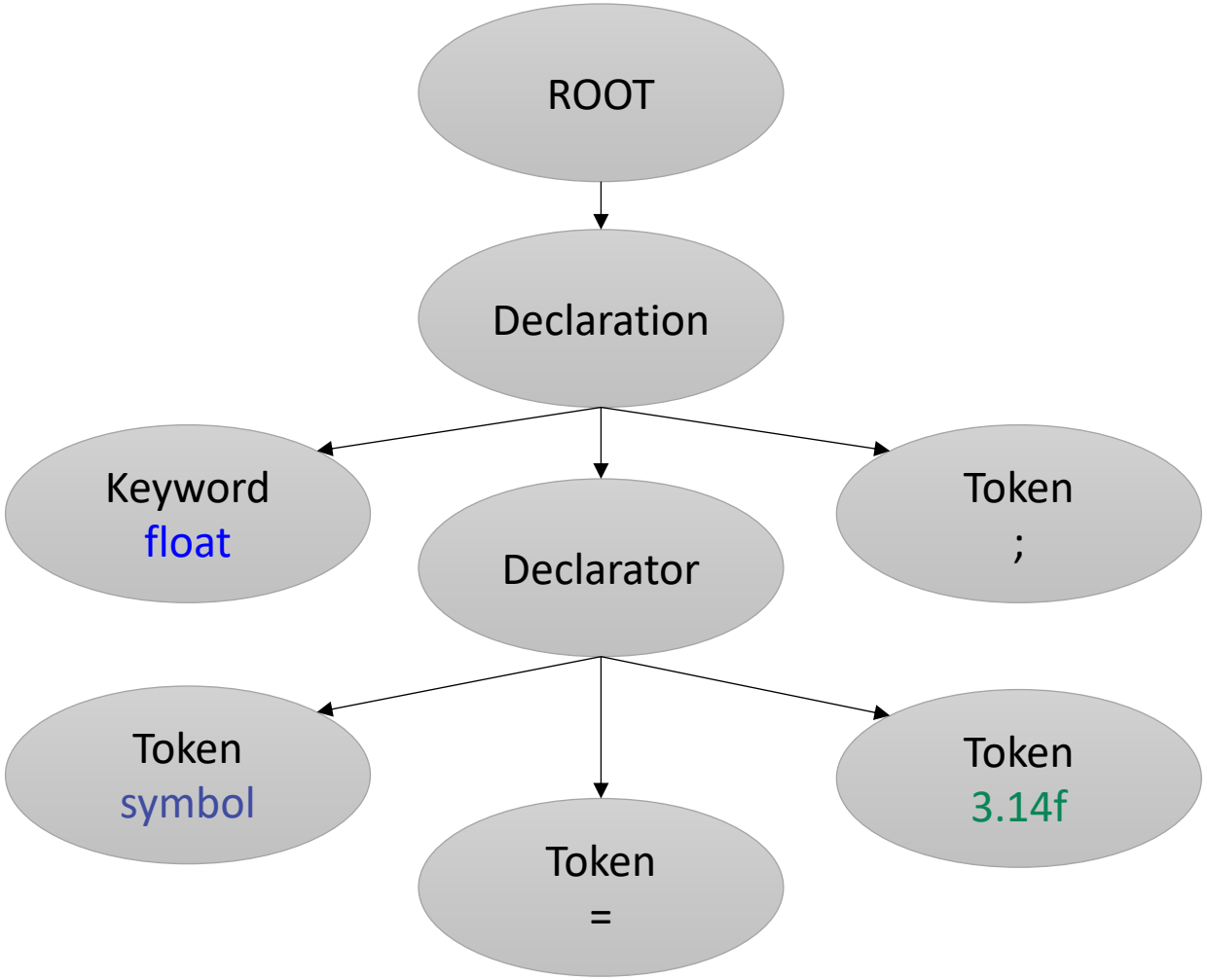
Visitor

```
struct Cat
{
  static int x;
};

Cat::x = 0;

int foo(int arg)
{
  float symbol = 3.14f;
  char x = 2;
  Cat dog { };

  return 0;
}
```

Трансляция

Visitor

```
struct Cat
{
  static int x;
};

Cat::x = 0;

int foo(int arg)
{
  float symbol = 3.14f;
  char x = 2;
  Cat dog { };

  return 0;
}
```



Построение символов

```
1:  struct Cat
2:  {
3:    static int x;
4:  };
5:
6:  Cat::x = 0;
7:  static void bar();
8:
9:  int foo(int arg)
10: {
11:   extern float symbol = 3.14f;
12:   static char x = 2;
13:   Cat dog { };
14:
15:   return 0;
16: }
```

Name (key)	Type	Declarator	Linkage	Storage Class
Cat	Class	1	external	
Cat::x	Builtin	3	external	static
bar	Function	7	internal	
foo	Function	9	external	
arg	Builtin	9	no	automatic
symbol	Builtin	11	external	automatic
x	Builtin	12	no	static
dog	Object	13	no	automatic

Семантический анализ

```
void check(int *ptr)
{
  int a = *ptr; // dereference
}

int bad()
{
  int *x = nullptr;
  check(x); // error
}
```

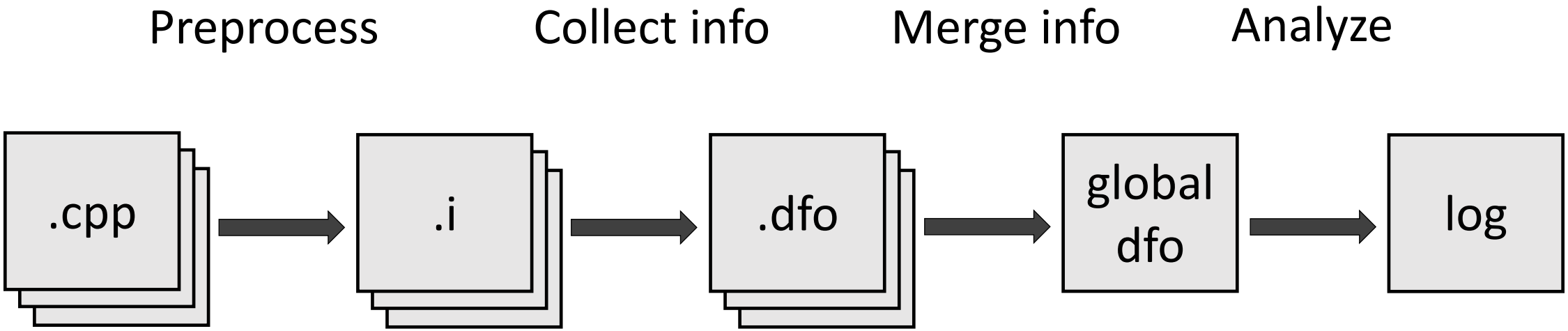
Data Flow
Symbolic Computation

Symbol | Data Flow Fact

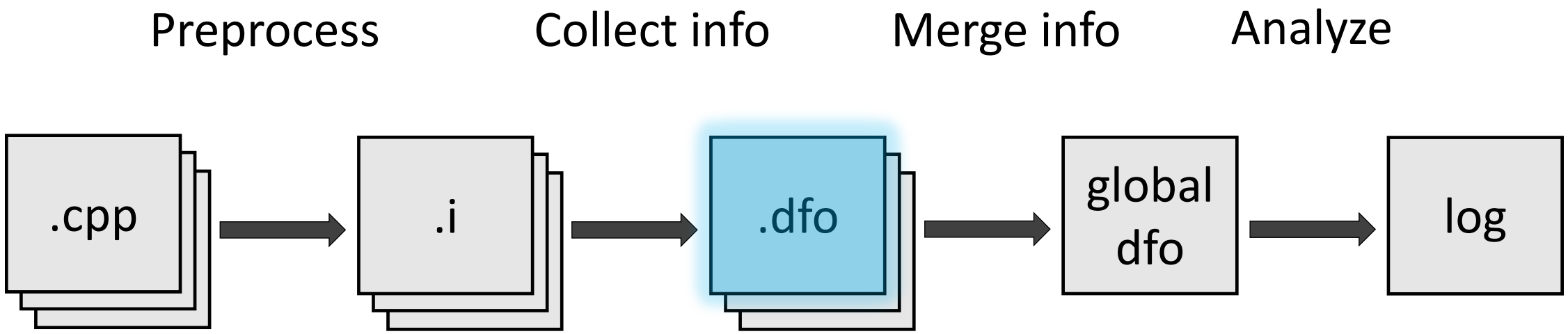
Fact[check][ptr] : <pointer, dereference, non-checked>

Решение

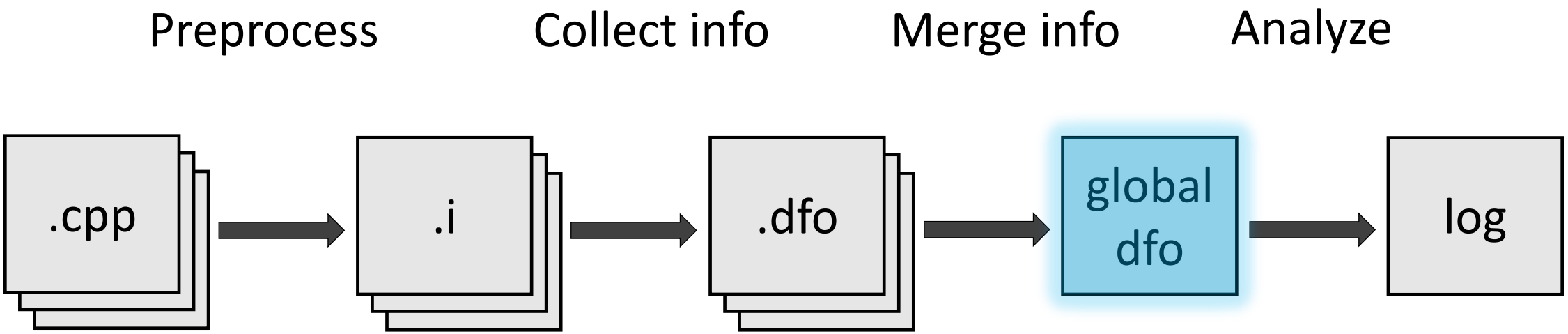
Предварительный анализ



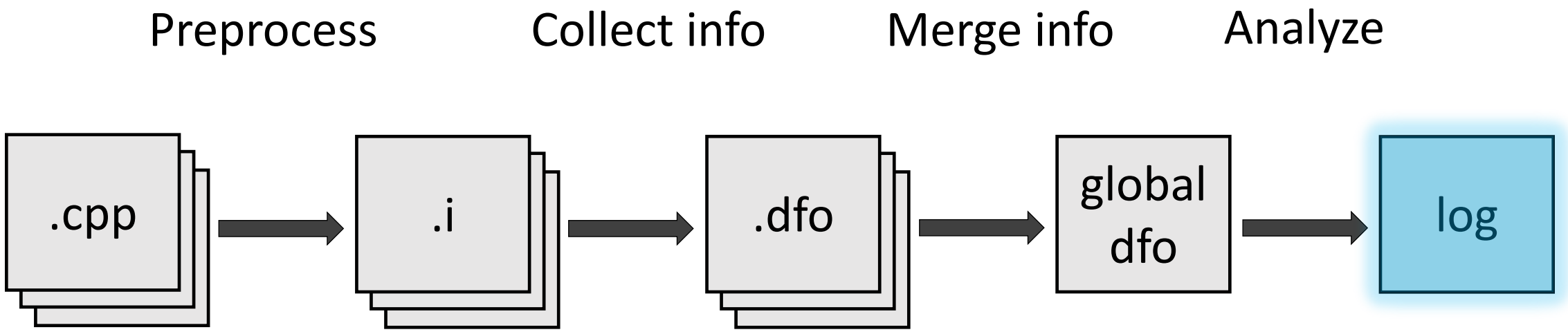
Предварительный анализ



Предварительный анализ



Предварительный анализ



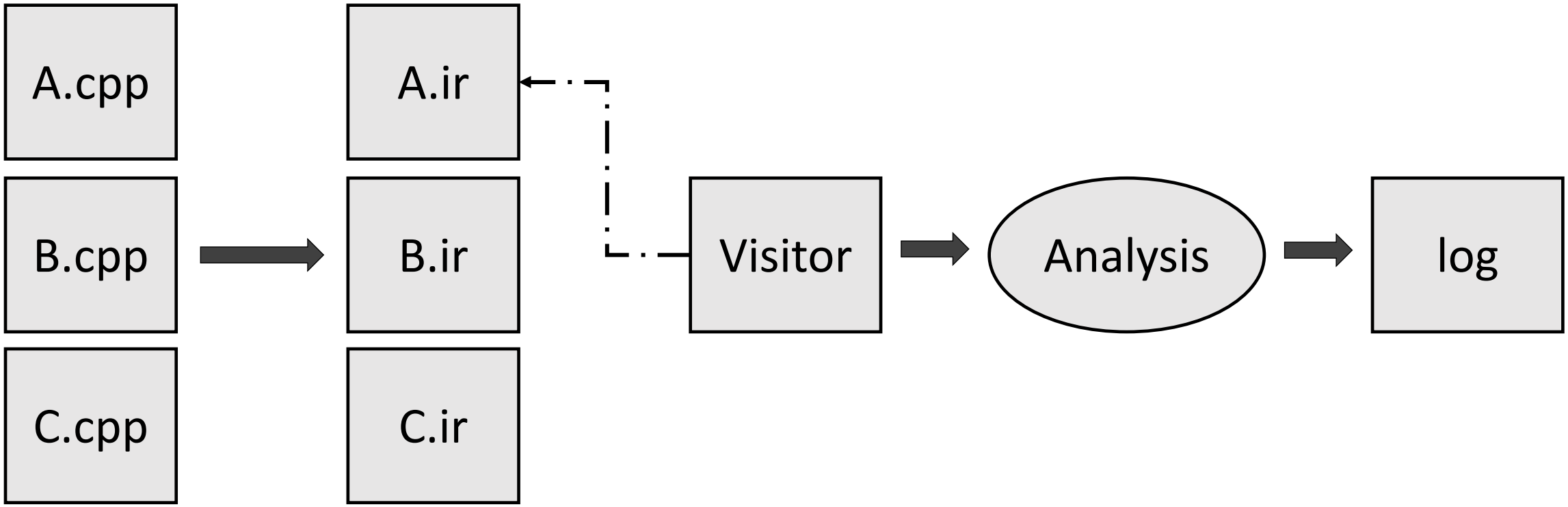
Вторая проблема

Вторая проблема

У нас нет промежуточного представления

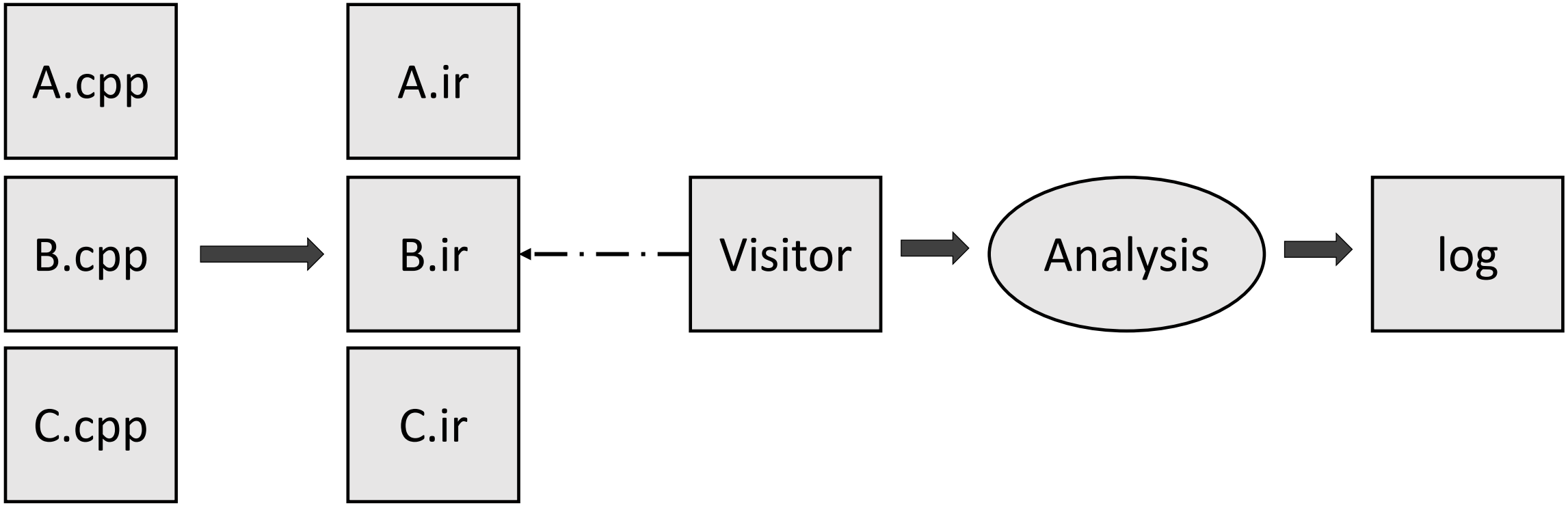
Вторая проблема

У нас нет промежуточного представления



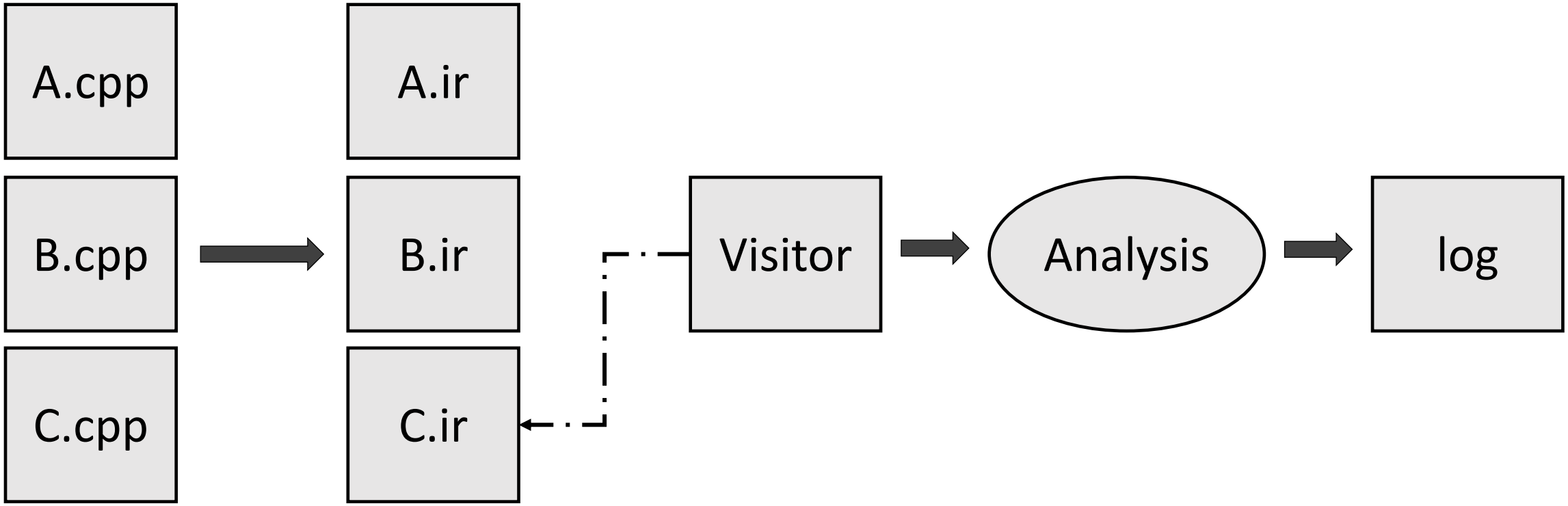
Вторая проблема

У нас нет промежуточного представления



Вторая проблема

У нас нет промежуточного представления

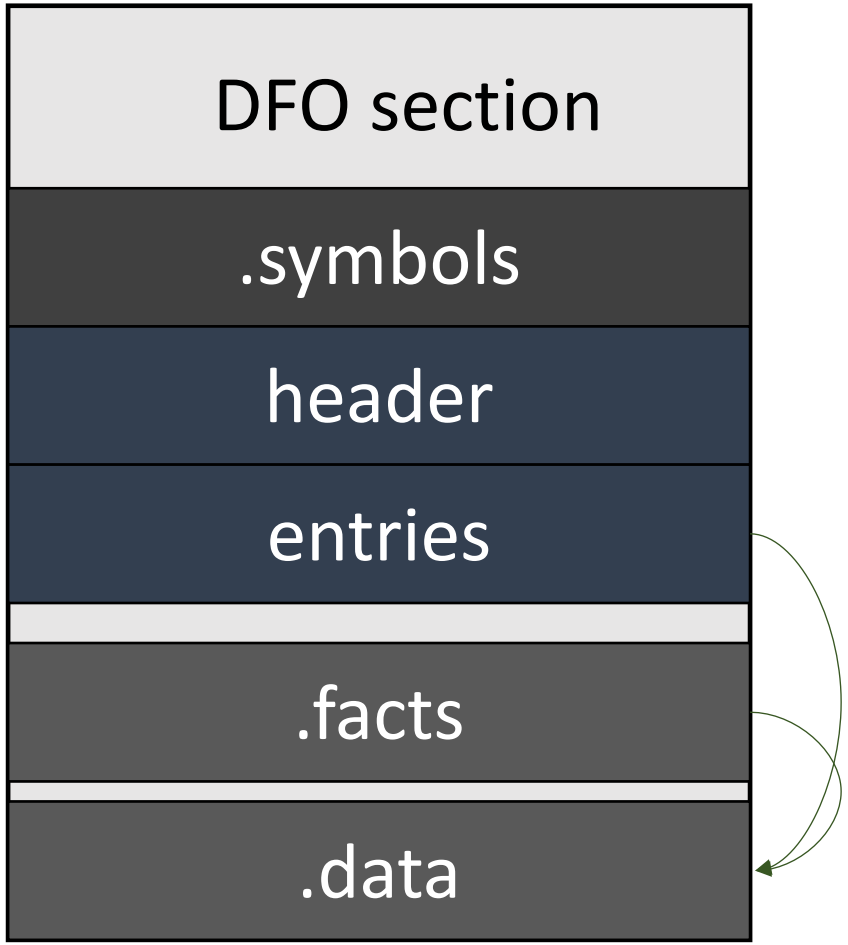


Решение

Сохранять только семантическую информацию

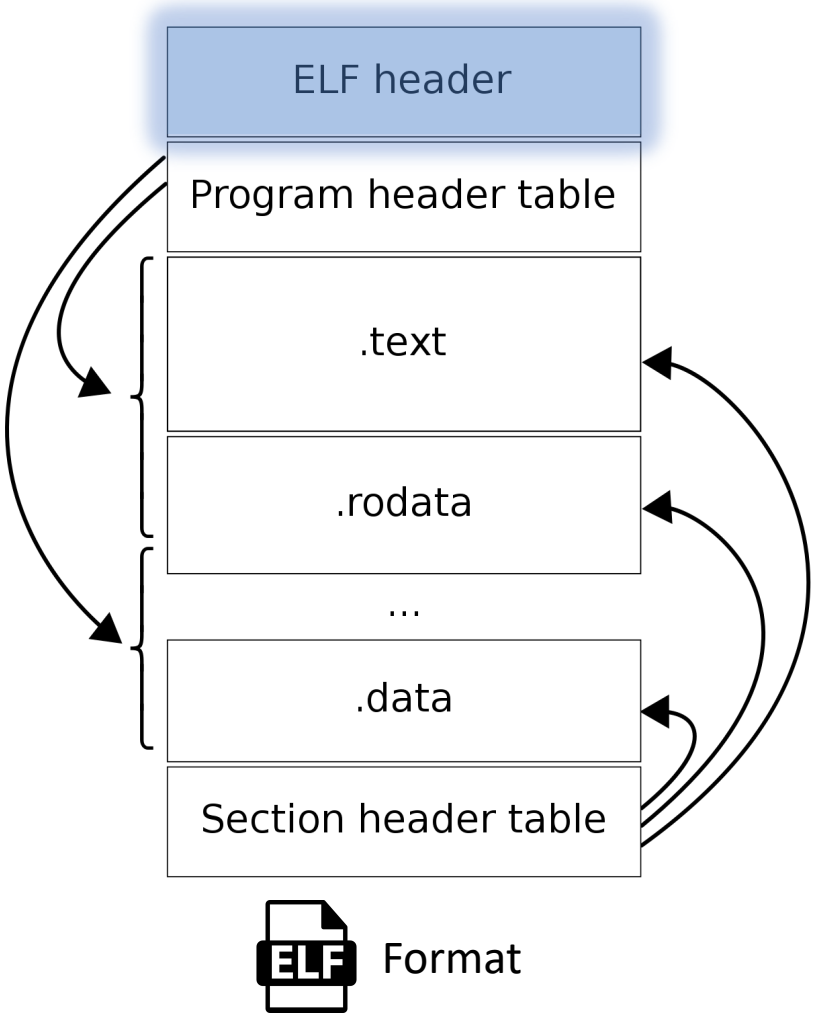
Symbol Info + Semantic Info + Source Location

Data Flow Object



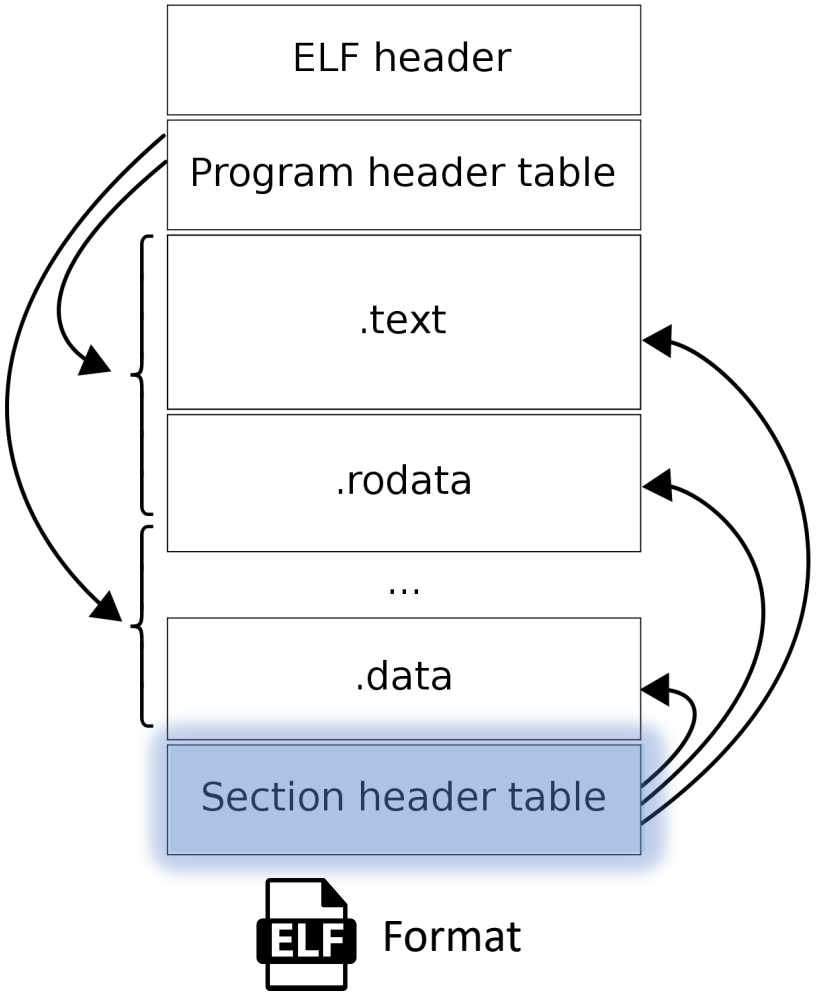
based

ELF Format



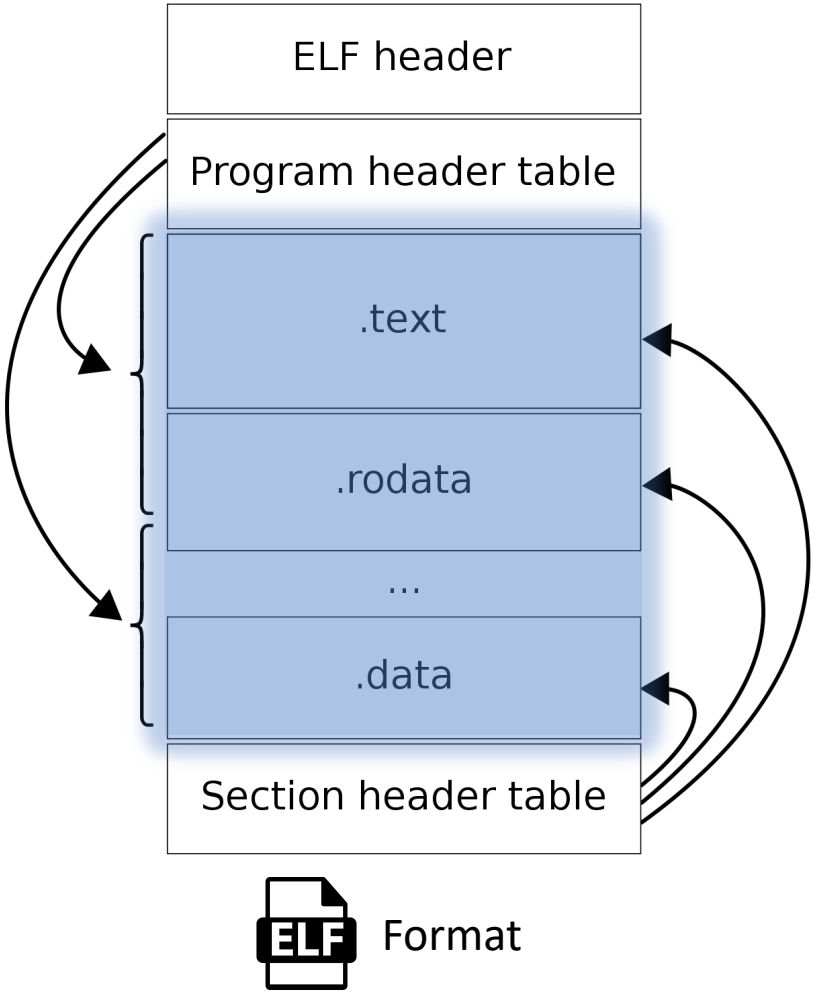

```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              REL (Relocatable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x0
  Start of program headers:          0 (bytes into file)
  Start of section headers:          688 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           0 (bytes)
  Number of program headers:         0
  Size of section headers:           64 (bytes)
  Number of section headers:         12
  Section header string table index: 1
There are 12 section headers, starting at offset 0x2b0:
```

ELF Format



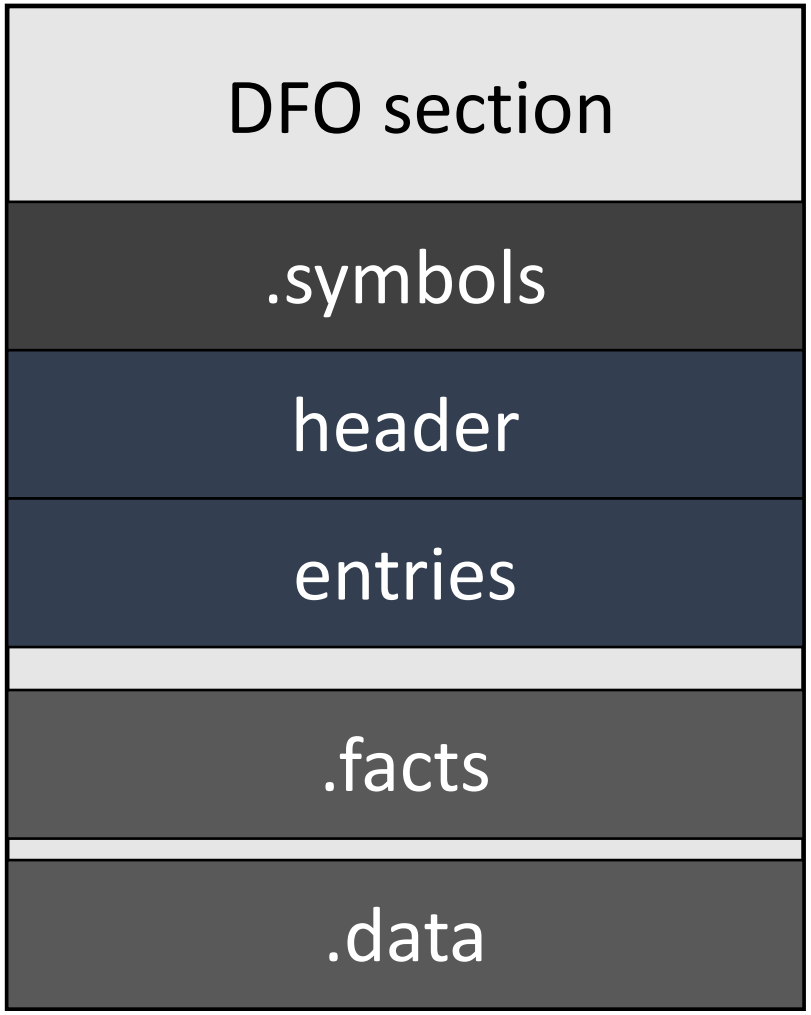

```
Section Headers:
  [Nr] Name              Type            Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            000000 000000 00      0   0  0
  [ 1] .strtab           STRTAB          0001b9 0000a3 00      0   0  1
  [ 2] .text             PROGBITS        000040 000016 00  AX  0   0 16
  [ 3] .rela.text        RELA            000188 000018 18     11   2  8
  [ 4] .data             PROGBITS        000058 000005 00  WA  0   0  4
  [ 5] .bss              NOBITS          00005d 000001 00  WA  0   0  1
  [ 6] .comment          PROGBITS        00005d 00002e 01  MS  0   0  1
  [ 7] .note.GNU-stack   PROGBITS        00008b 000000 00      0   0  1
  [ 8] .eh_frame         X86_64_UNWIND   000090 000038 00   A  0   0  8
  [ 9] .rela.eh_frame    RELA            0001a0 000018 18     11   8  8
  [10] .llvm_addrsig     LLVM_ADDRSIG    0001b8 000001 00   E 11   0  1
  [11] .symtab           SYMTAB          0000c8 0000c0 18      1   6  8
```

ELF Format

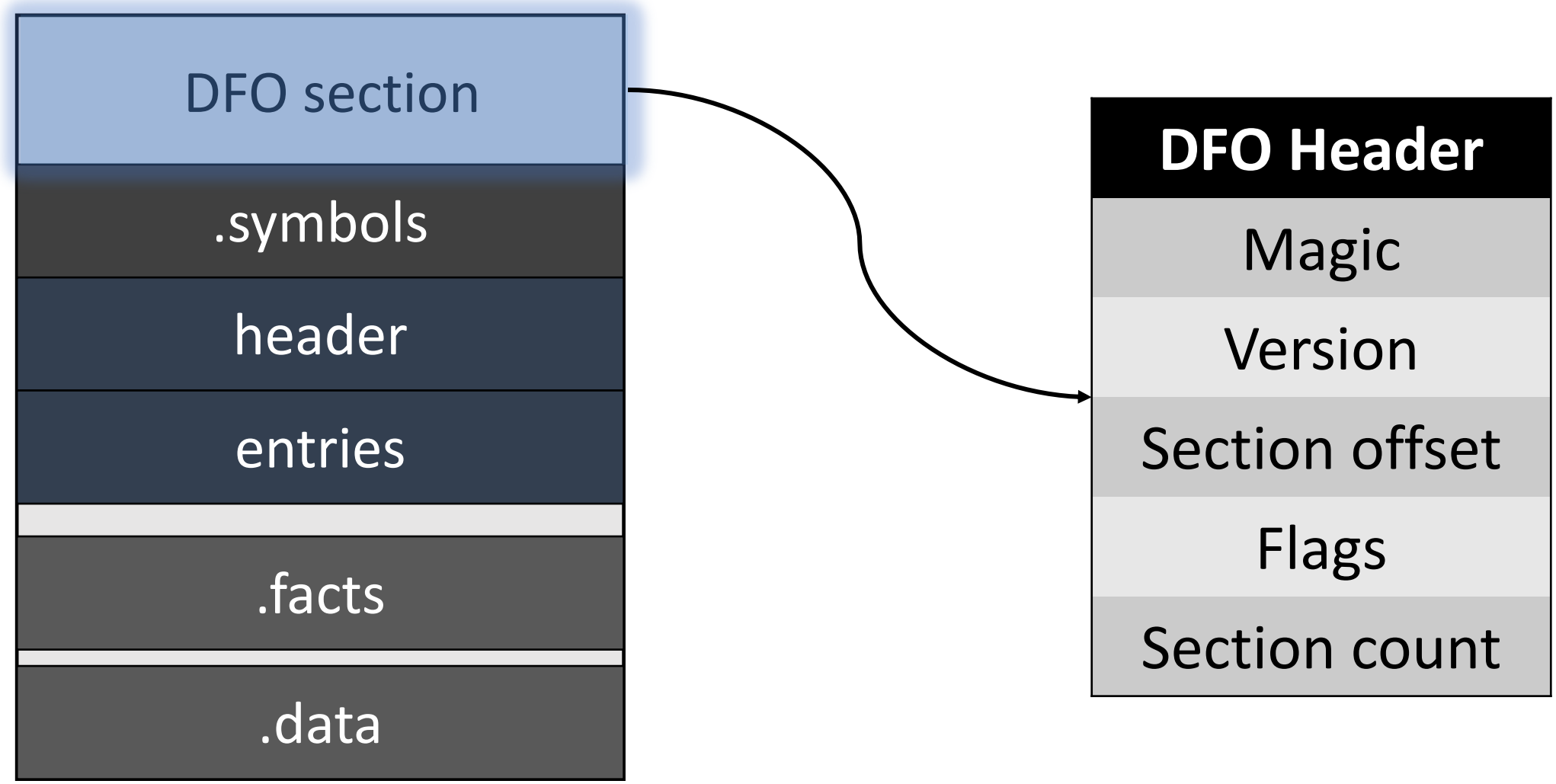



```
Symbol table '.symtab' contains 8 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     0: 0000000000000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 0000000000000000     0 FILE    LOCAL  DEFAULT  ABS sym.cpp
     2: 0000000000000004     1 OBJECT  LOCAL  DEFAULT    4 foo(int)::x
     3: 0000000000000000     1 OBJECT  LOCAL  DEFAULT    5 foo(int)::dog
     4: 0000000000000000     4 OBJECT  LOCAL  DEFAULT    4 foo(int)::symbol
     5: 0000000000000000     0 SECTION LOCAL  DEFAULT    2 .text
     6: 0000000000000000    22 FUNC    GLOBAL DEFAULT    2 foo(int)
     7: 0000000000000000     0 NOTYPE  GLOBAL DEFAULT  UND Cat::x
```

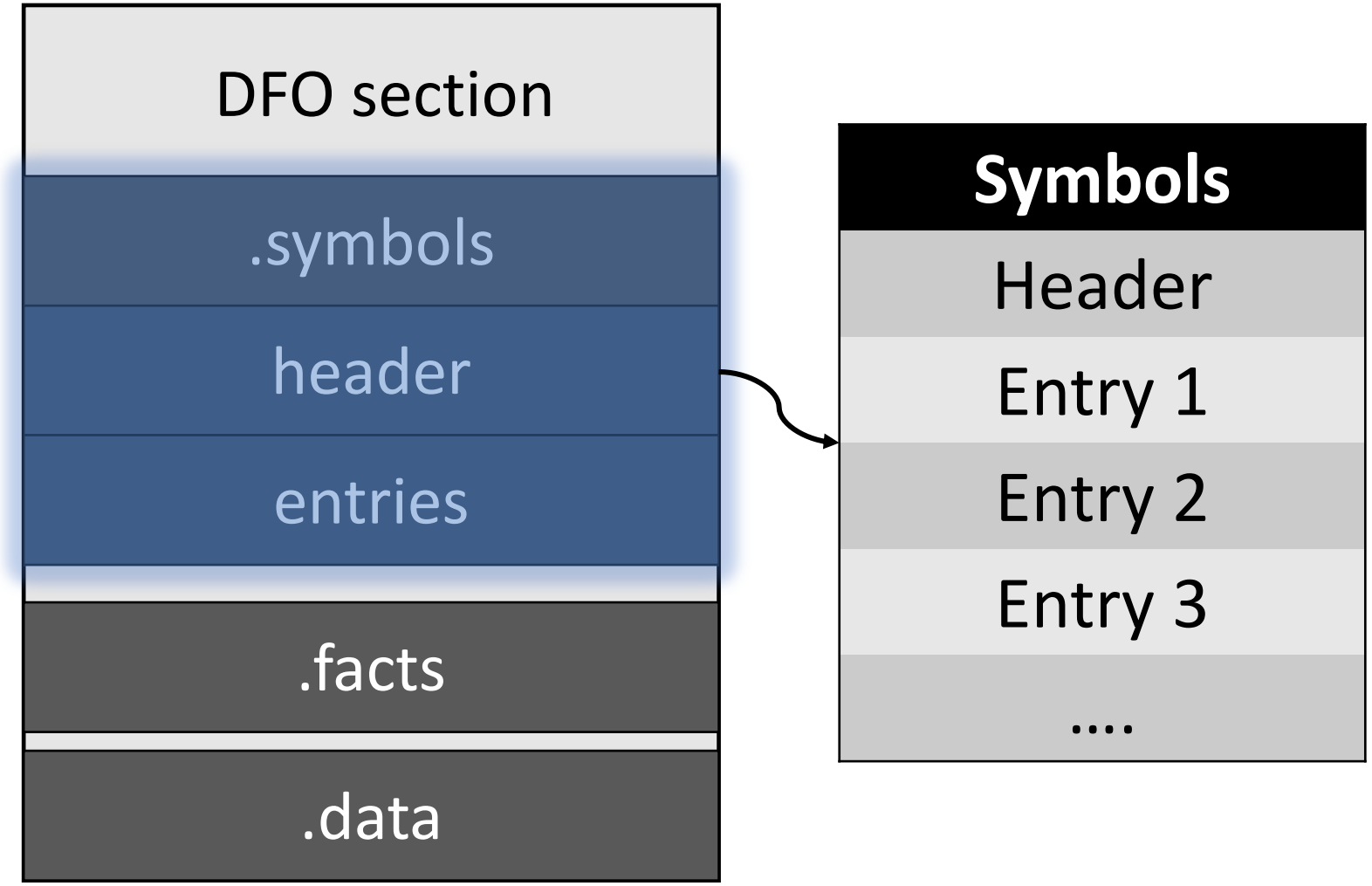
Data Flow Object



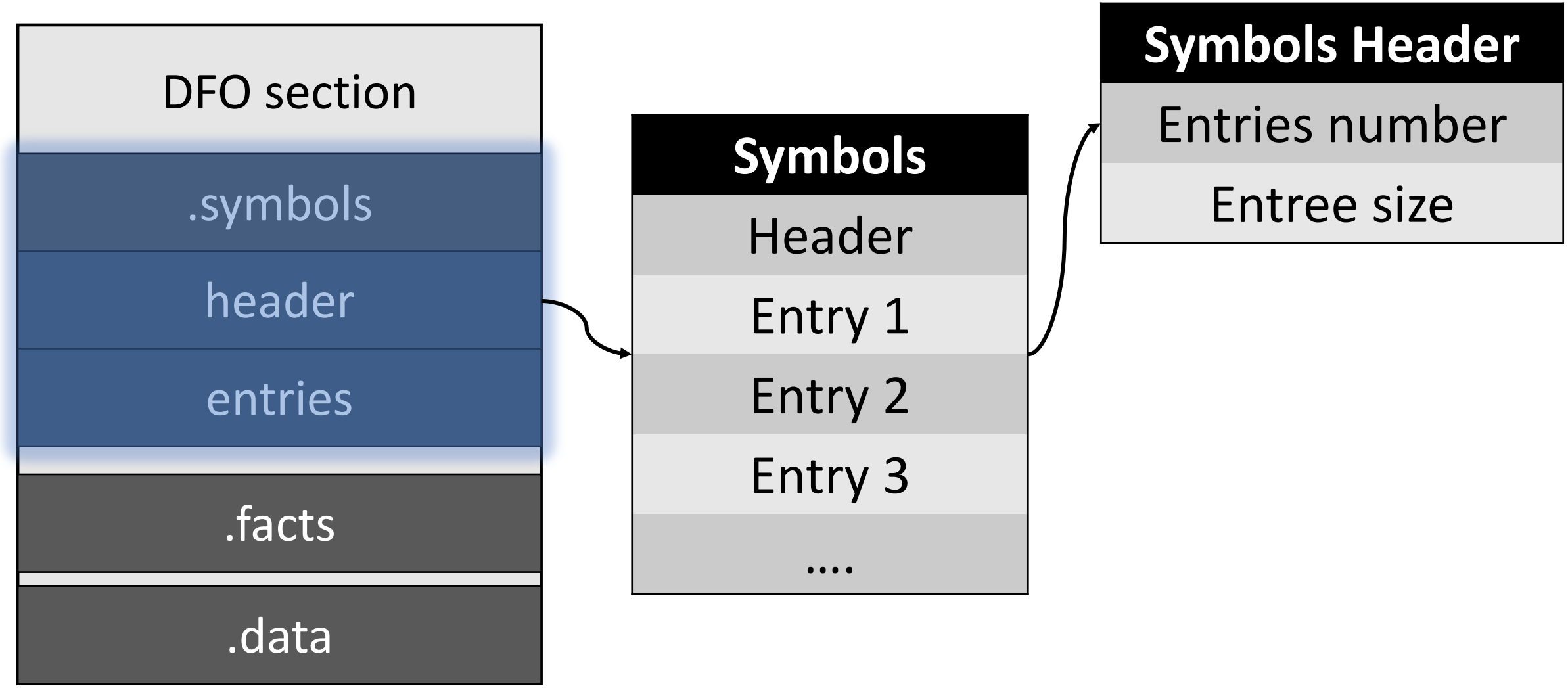
Data Flow Object



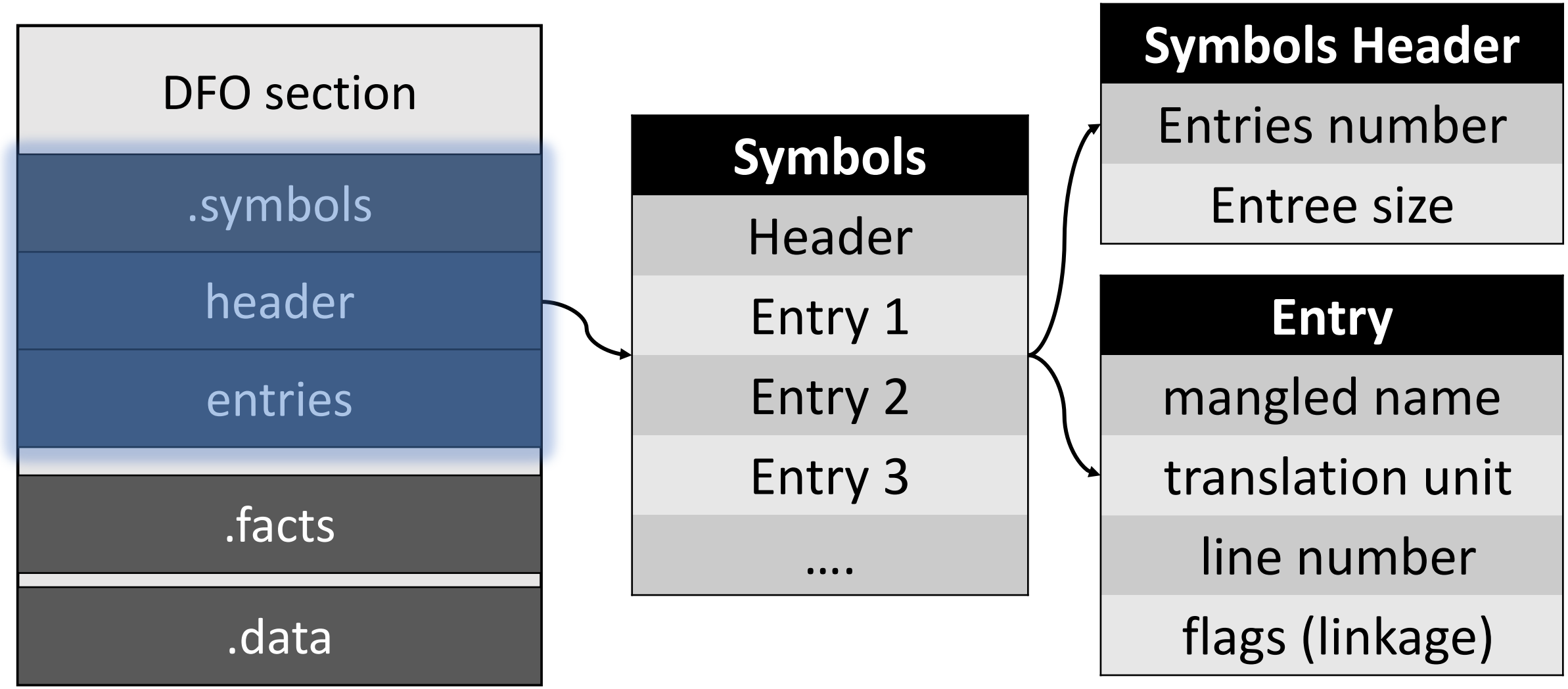
Data Flow Object



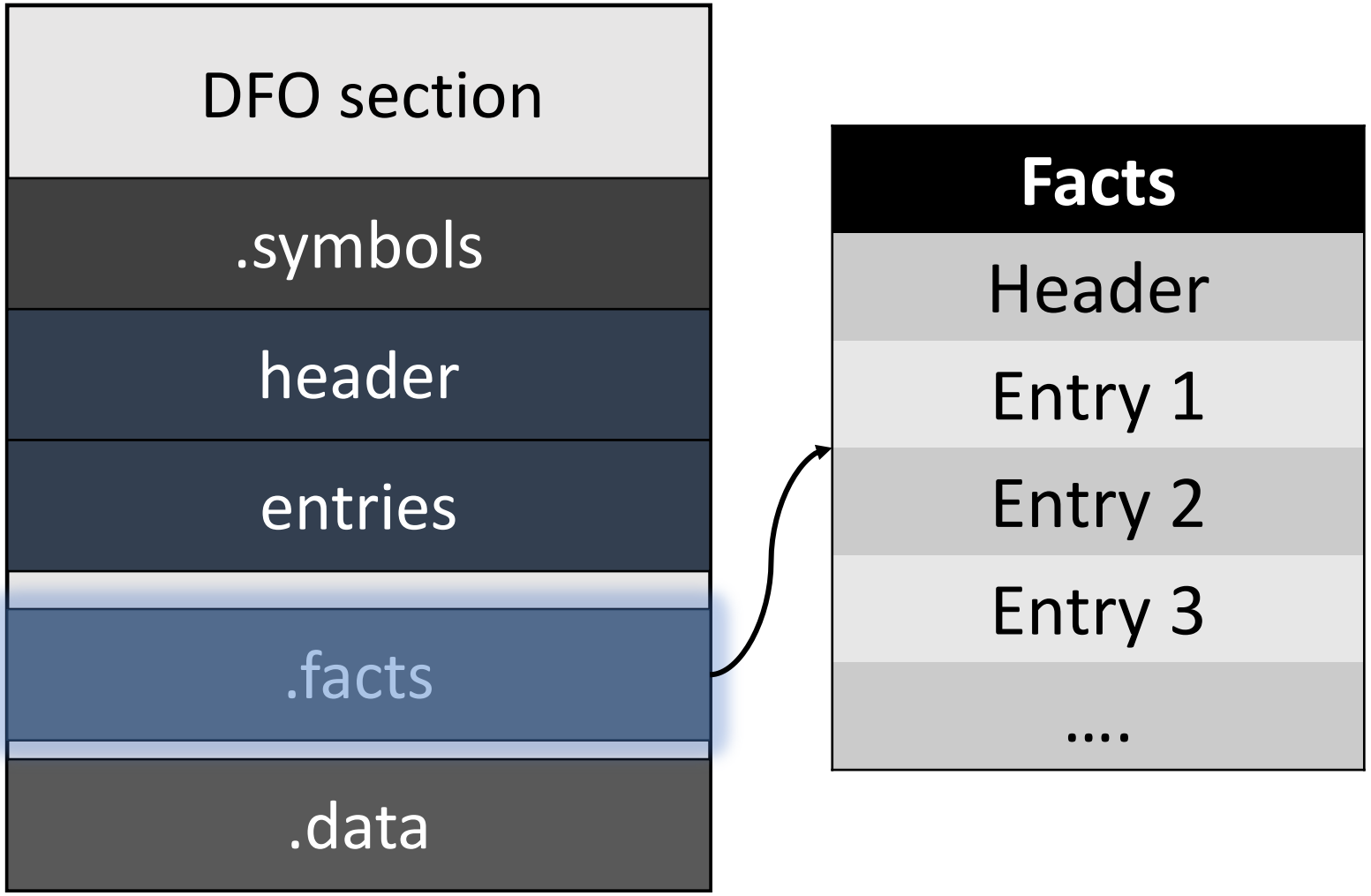
Data Flow Object



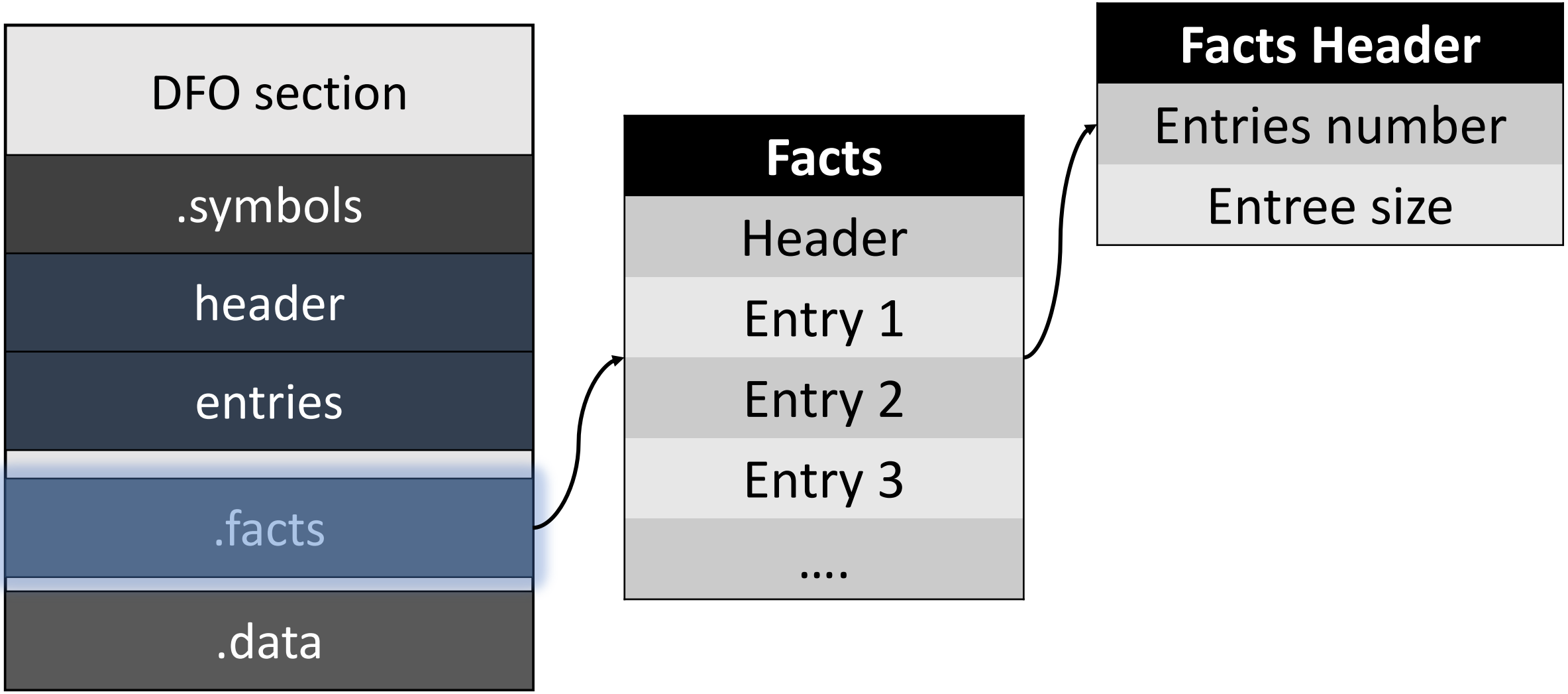
Data Flow Object



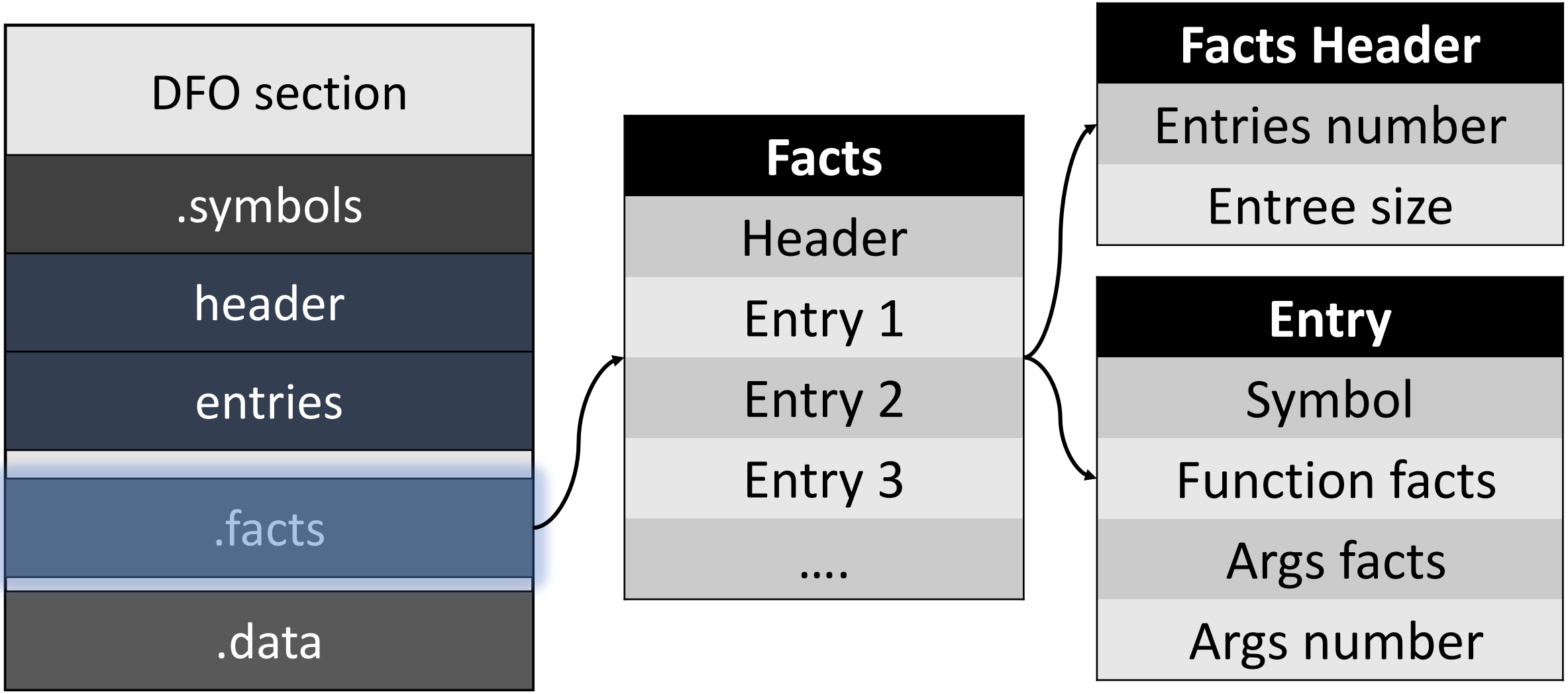
Data Flow Object



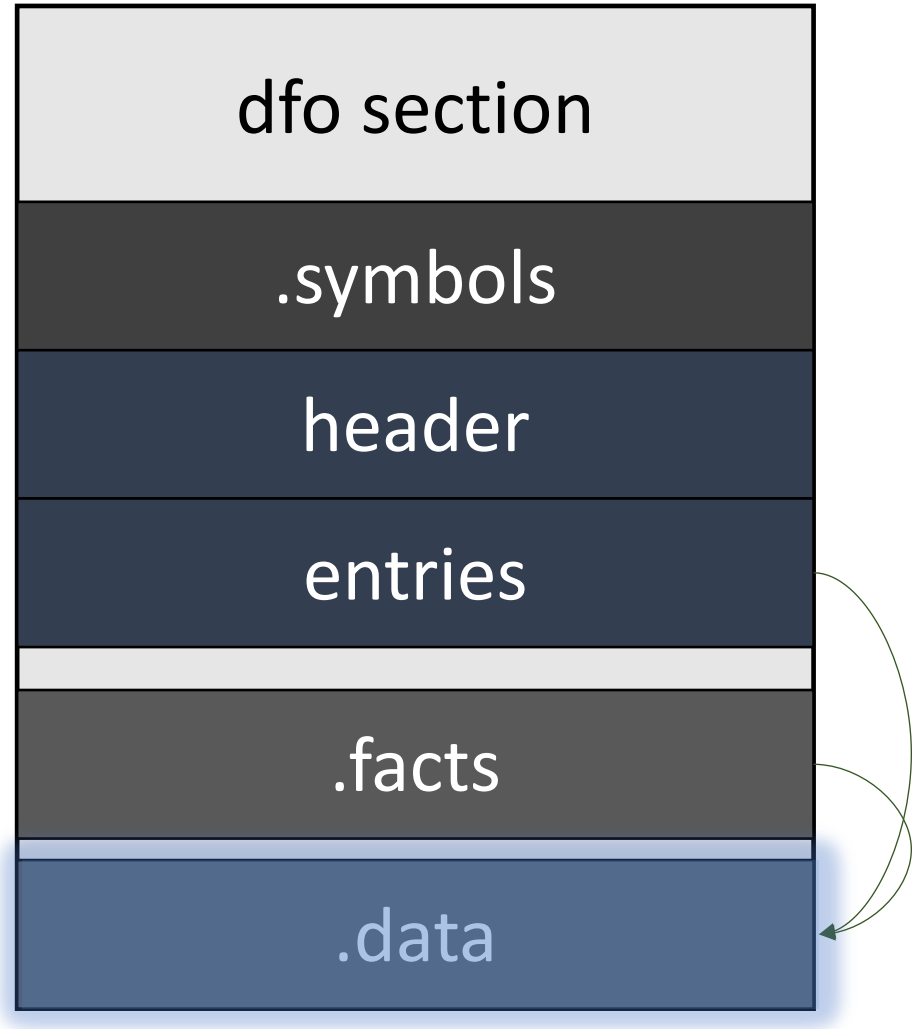
Data Flow Object



Data Flow Object



Data Flow Object




```
-> % hexdump -C simple.PVS-Studio.dfo
00000000  7f 44 46 4f 00 00 00 00  01 00 00 00 00 00 00 00  |.DFO............|
00000010  20 00 00 00 00 00 00 00  03 00 00 00 00 00 00 00  | ...............|
00000020  64 01 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |d...............|
00000030  00 00 00 00 00 00 00 00  48 00 00 00 00 00 00 00  |........H.......|
00000040  30 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  |0...............|
00000050  20 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  | ...............|
00000060  0a 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00  |................|
00000070  02 00 00 00 00 00 00 00  6c 01 00 00 00 00 00 00  |........l.......|
00000080  01 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00000090  a0 00 00 00 00 00 00 00  30 00 00 00 00 00 00 00  |........0.......|
000000a0  01 00 00 00 00 00 00 00  20 00 00 00 00 00 00 00  |........ .......|
000000b0  00 00 00 00 00 00 00 00  28 00 00 00 00 00 00 00  |........(.......|
000000c0  30 00 00 00 00 00 00 00  02 00 00 00 00 00 00 00  |0...............|
000000d0  73 01 00 00 00 00 00 00  02 00 00 00 00 00 00 00  |s...............|
000000e0  00 00 00 00 00 00 00 00  f8 00 00 00 00 00 00 00  |................|
000000f0  79 01 00 00 00 00 00 00  61 64 64 2d 46 69 69 2d  |y.......add-Fii-|
00000100  69 00 44 3a 5c 52 65 70  6f 5c 74 65 73 74 50 72  |i.D:\Repo\testPr|
00000110  65 73 61 5c 73 69 6d 70  6c 65 2e 63 70 70 00 00  |esa\simple.cpp..|
```

Выравнивание

```
Align(x) = alignof(decltype(x))
Size(x) = sizeof(x)
```

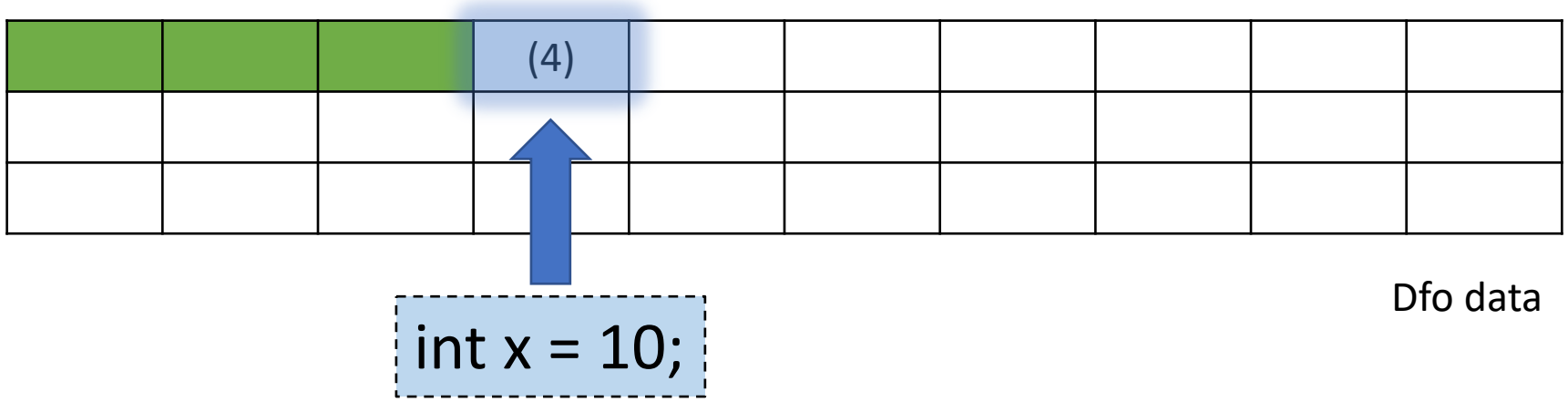
Выравнивание

```
Align(x) = alignof(decltype(x))
Size(x) = sizeof(x)
```

Dfo data

Выравнивание

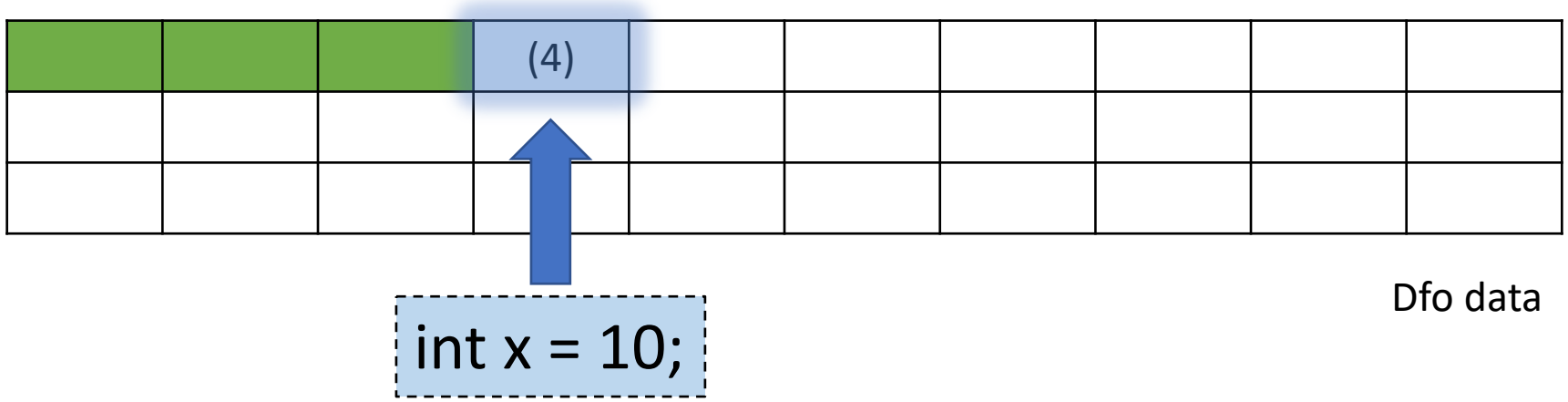
```
Align(x) = alignof(decltype(x))
Size(x) = sizeof(x)
```




```
additionalBytes = (align - data.size() % align) % align;
```

Выравнивание

```
Align(x) = alignof(decltype(x))
Size(x) = sizeof(x)
```




```
additionalBytes = (align - data.size() % align) % align =
= (4 - 3 % 4) % 4 = 1 byte;
```

Выравнивание

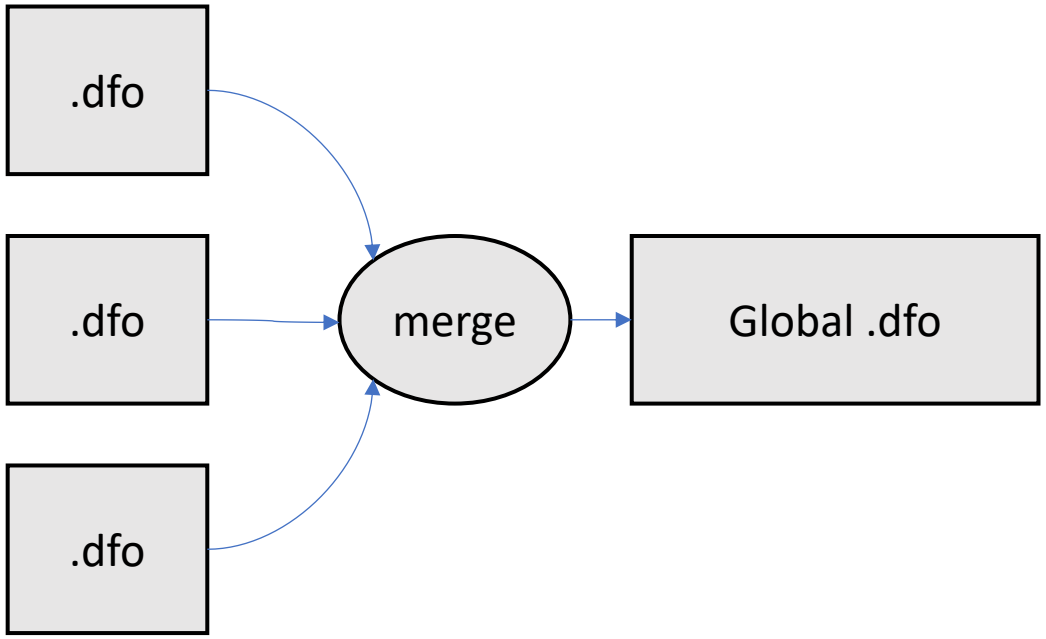
```
Align(x) = alignof(decltype(x))
Size(x) = sizeof(x)
```

				00	00	00	0A		

Dfo data

```
additionalBytes = (align - data.size() % align) % align =
= (4 - 3 % 4) % 4 = 1 byte;
```

Symbol Resolution



Условия:

1. Программа компилируется
2. На вход анализатора подаются только файлы, принадлежащие проекту

Symbol Resolution

A.cpp

```
std::string Revert(std::string value);
```

→ Ничего не собрано

B.cpp

```
std::string Revert(std::string value)
{
  ....
}
```

→ Revert-FQstd::stringQstd::string

Symbol Resolution

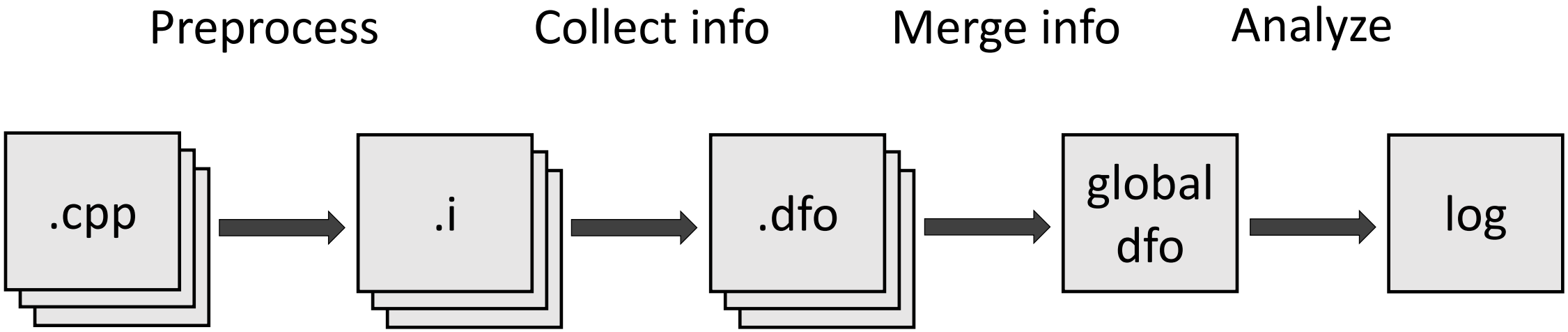
```
A.cpp
std::string Revert(std::string value)
{
   ....
}
```

→ Revert-FQstd::stringQstd::string

```
B.cpp
std::string Revert(std::string value)
{
   ....
}
```

→ Игнорируется при совпадении атрибутов

Глубокий анализ



Глубокий анализ

```
int f1(int *ptr);

int main()
{
  return f1(nullptr);
}
```

```
int f2(int *ptr);

int f1(int *ptr)
{
  return f2(ptr);
}
```

```
int f2(int *ptr)
{
  return *ptr;
}
```

Глубокий анализ

```
int f1(int *ptr);

int main()
{
  return f1(nullptr);
}
```

```
int f2(int *ptr);

int f1(int *ptr)
{
  return f2(ptr);
}
```

```
int f2(int *ptr)
{
  return *ptr;
}
```

Глубокий анализ

```
int f1(int *ptr);

int main()
{
  return f1(nullptr);
}
```

```
int f2(int *ptr);

int f1(int *ptr)
{
  return f2(ptr);
}
```

```
int f2(int *ptr)
{
  return *ptr;
}
```

Глубокий анализ

```
int f1(int *ptr);

int main()
{
  return f1(nullptr);
}
```

```
int f2(int *ptr);

int f1(int *ptr)
{
  return f2(ptr);
}
```

```
int f2(int *ptr)
{
  return *ptr;
}
```

Глубокий анализ

Шаг 1 – Обнаружить факт, что f2 выполняет разыменовывание аргумента

dereference

```
int f1(int *ptr);

int main()
{
  return f1(nullptr);
}
```

```
int f2(int *ptr);

int f1(int *ptr)
{
  return f2(ptr);
}
```

```
int f2(int *ptr)
{
  return *ptr;
}
```

Глубокий анализ

Шаг 1 – Обнаружить факт, что f2 выполняет разыменовывание аргумента

Шаг 2 – Обнаружить факт, что f1 выполняет разыменовывание аргумента

dereference

```
int f1(int *ptr);

int main()
{
  return f1(nullptr);
}
```

dereference

```
int f2(int *ptr);

int f1(int *ptr)
{
  return f2(ptr);
}
```

```
int f2(int *ptr)
{
  return *ptr;
}
```

Глубокий анализ

Шаг 1 – Обнаружить факт, что f2 выполняет разыменовывание аргумента

Шаг 2 – Обнаружить факт, что f1 выполняет разыменовывание аргумента

Шаг 3 – Обнаружить что вызов f1 приведет к ошибке

dereference

```
int f1(int *ptr);

int main()
{
  return f1(nullptr);
}
```

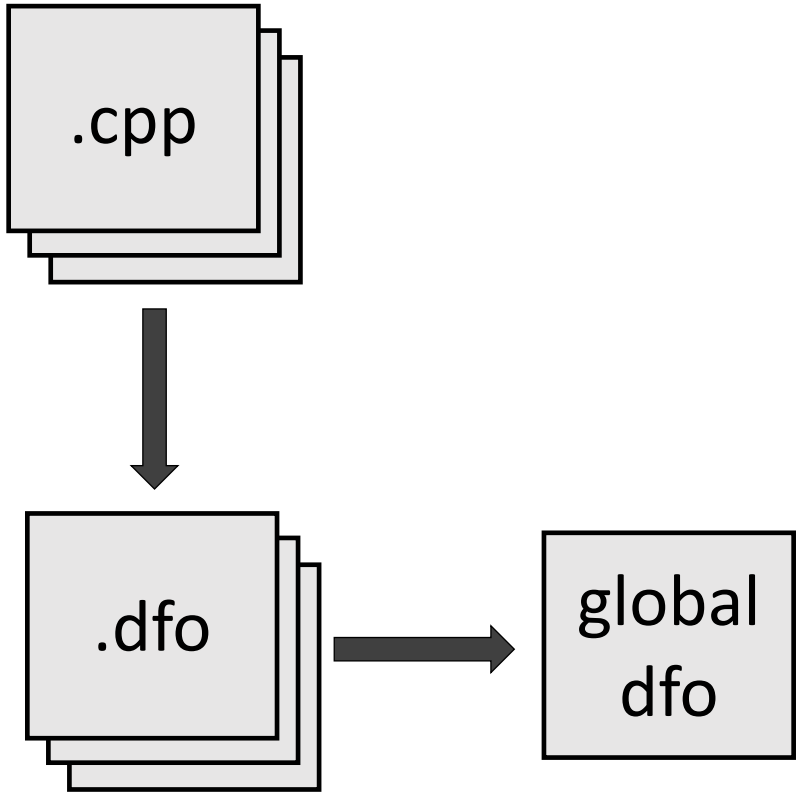
dereference

```
int f2(int *ptr);

int f1(int *ptr)
{
  return f2(ptr);
}
```

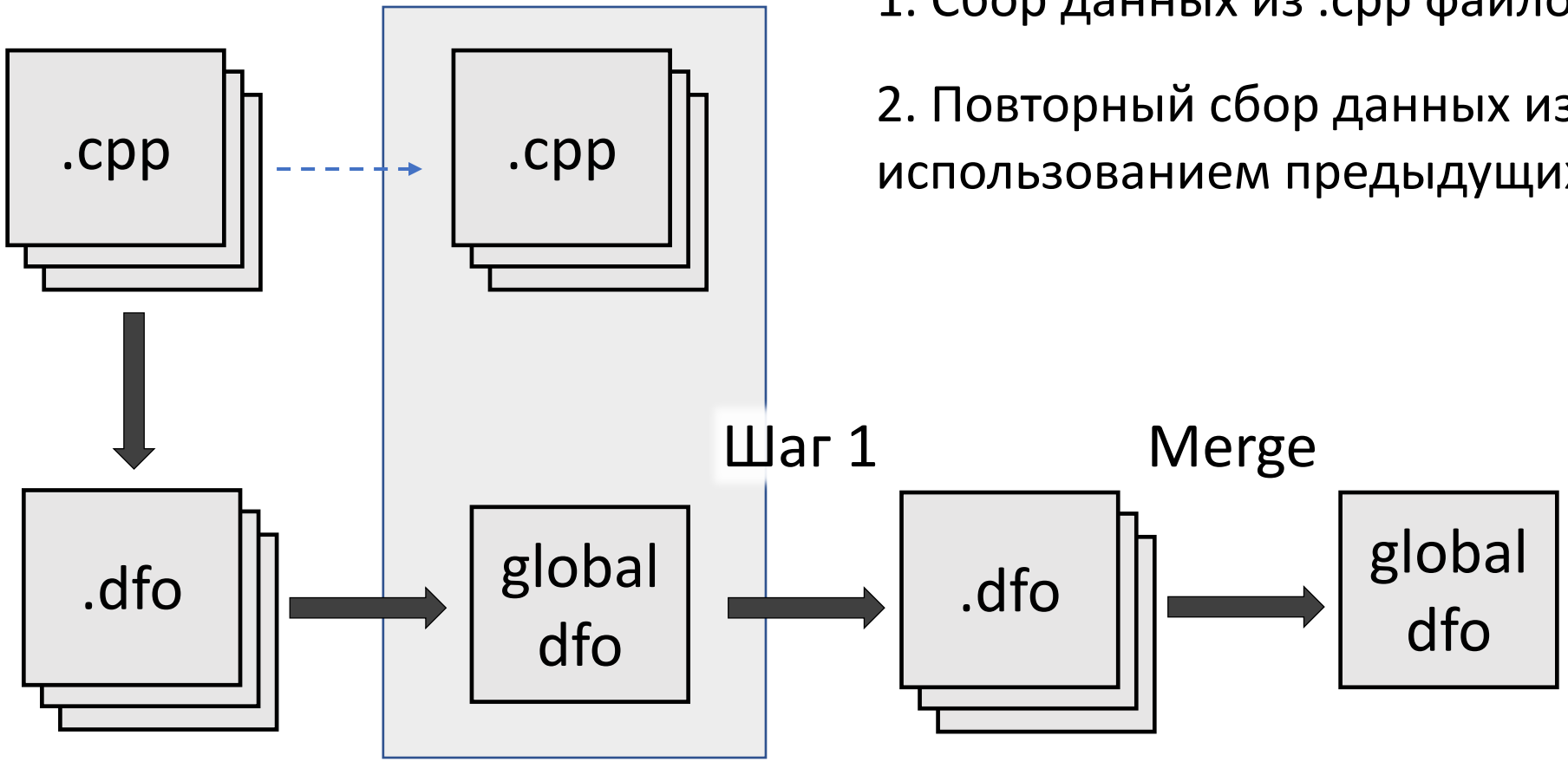
```
int f2(int *ptr)
{
  return *ptr;
}
```

Глубокий анализ

1. Сбор данных из .cpp файлов в global dfo



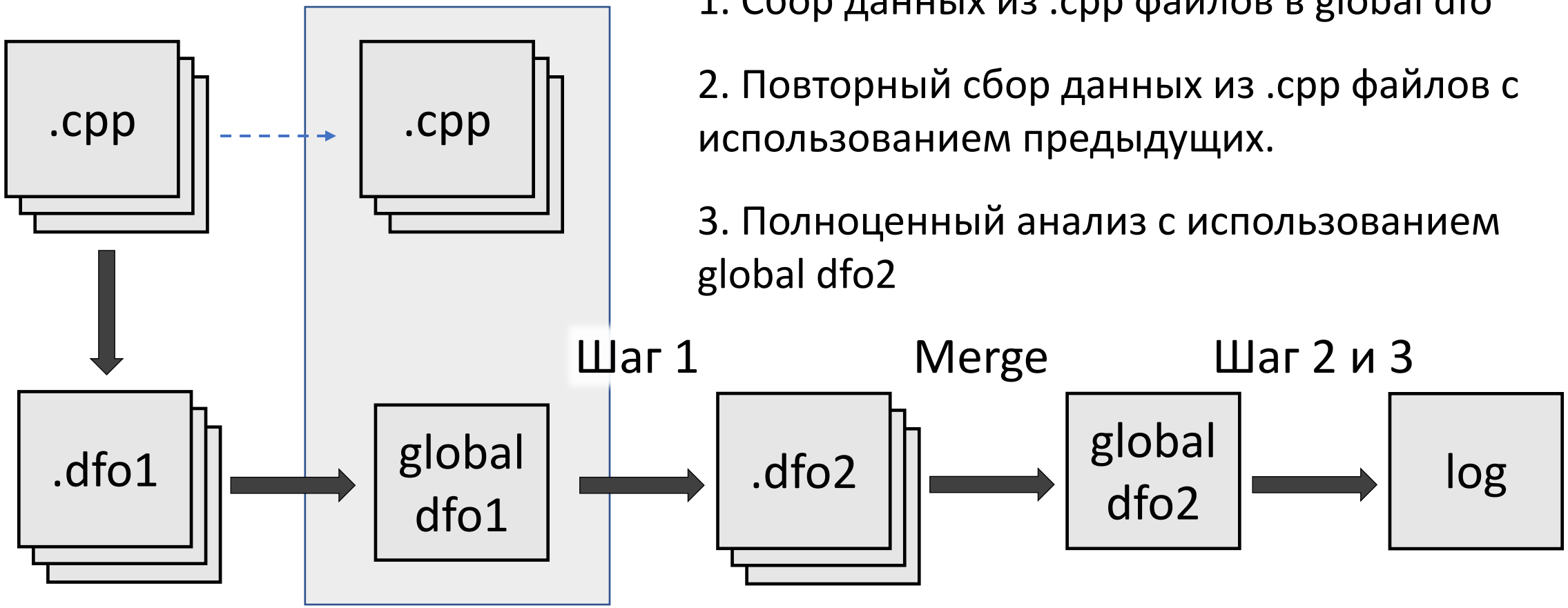
Глубокий анализ



1. Сбор данных из .cpp файлов в global dfo

2. Повторный сбор данных из .cpp файлов с использованием предыдущих.

Глубокий анализ



1. Сбор данных из .cpp файлов в global dfo

2. Повторный сбор данных из .cpp файлов с использованием предыдущих.

3. Полноценный анализ с использованием global dfo2

Анализ составных проектов

```
project(multylib)
add_library(lib1 A.cpp B.cpp)
add_library(lib2 B.cpp)
```

Анализ составных проектов

```
compile_commands.json:
[
  {
    "file": "....\\A.cpp",
    "command": "clang-cl.exe ....\\A.cpp -m64 /Zi /Ob0 /Od /RTC1 -MDd -std:c++latest",
    "directory": "…\\projectDir"
  },
  {
    "file": "....\\B.cpp",
    "command": "clang-cl.exe ....\\B.cpp -m64 /Zi /Ob0 /Od /RTC1 -MDd -std:c++latest",
    "directory": "…\\projectDir "
  },
  {
    "file": "....\\B.cpp",
    "command": "clang-cl.exe ....\\B.cpp -m64 /Zi /Ob0 /Od /RTC1 -MDd -std:c++latest",
    "directory": "....\\projectDir "
  }
]
```

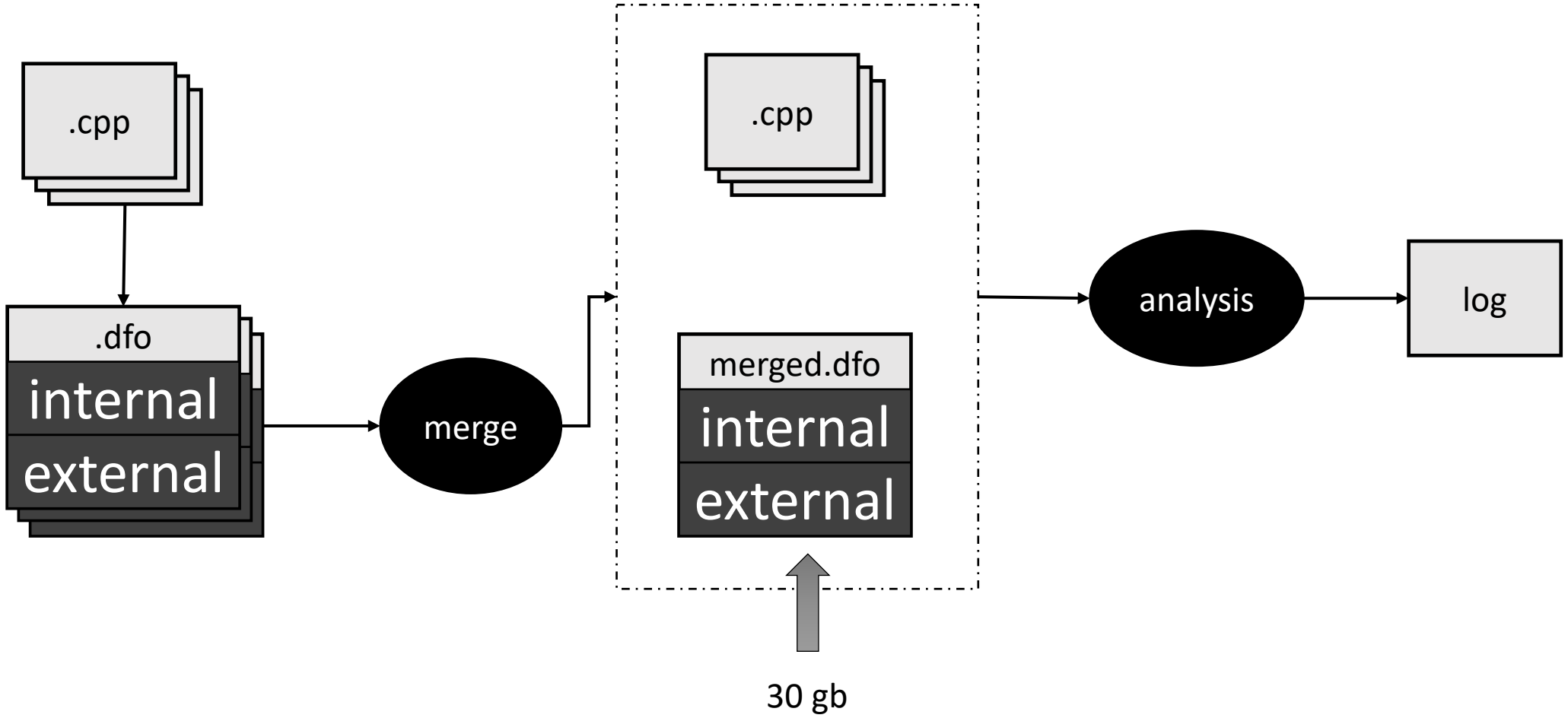
Анализ составных проектов

cmake 3.20

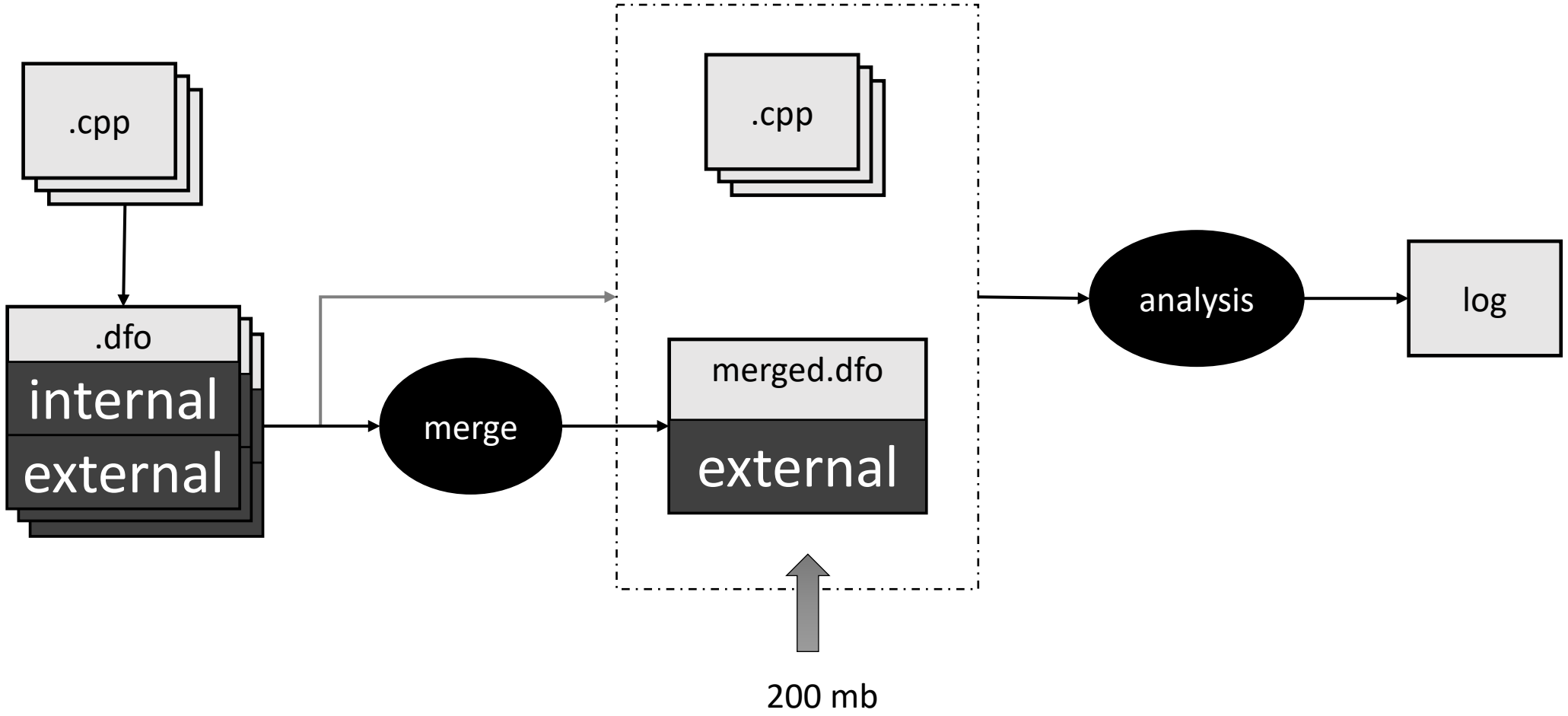
```
CMakeLists.txt:
....
project(multilib)
....

set(CMAKE_EXPORT_COMPILE_COMMANDS FALSE) #disable generation for all targets

add_library(lib1 A.cpp B.cpp)
add_library(lib2 B.cpp)

#enable generatrion for lib2
set_property(TARGET lib2 PROPERTY EXPORT_COMPILE_COMMANDS TRUE)
```

Оптимизации анализа

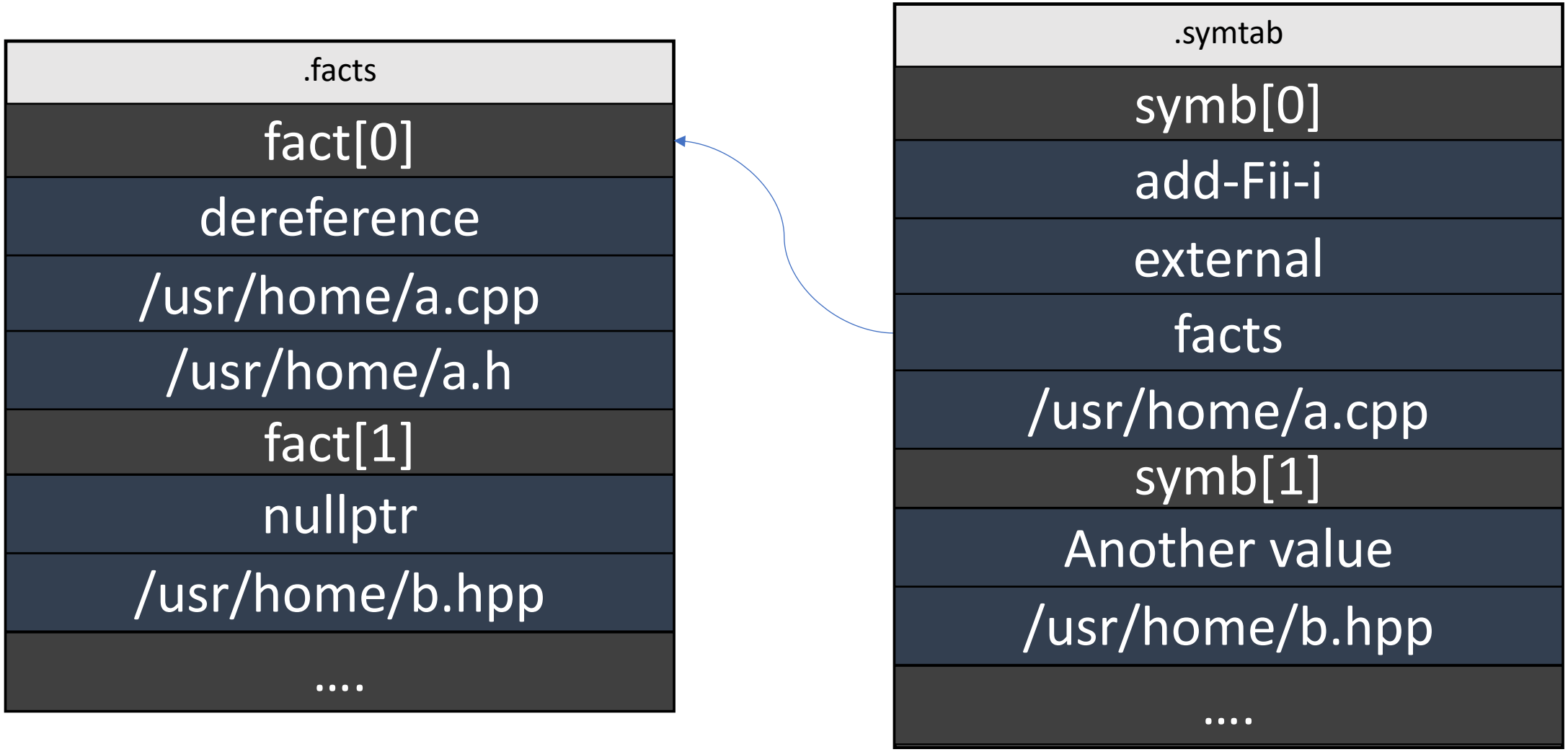
Раздельный анализ internal и external символов



Раздельный анализ internal и external символов

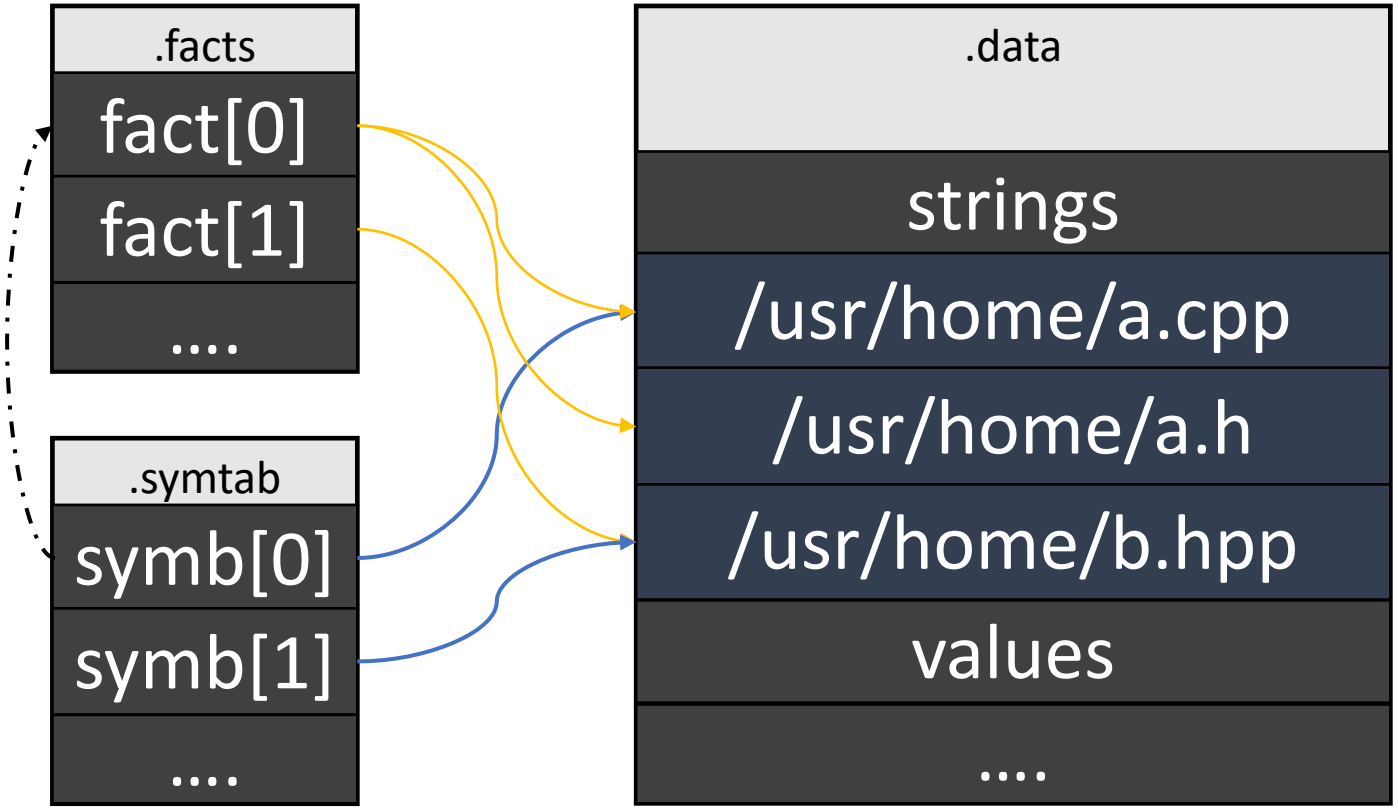


Интернирование



Интернирование

Key	Value
/usr/home/a.cpp	DataOffset + 0
/usr/home/a.h	DataOffset + 16
/usr/home/b.hpp	DataOffset + 33

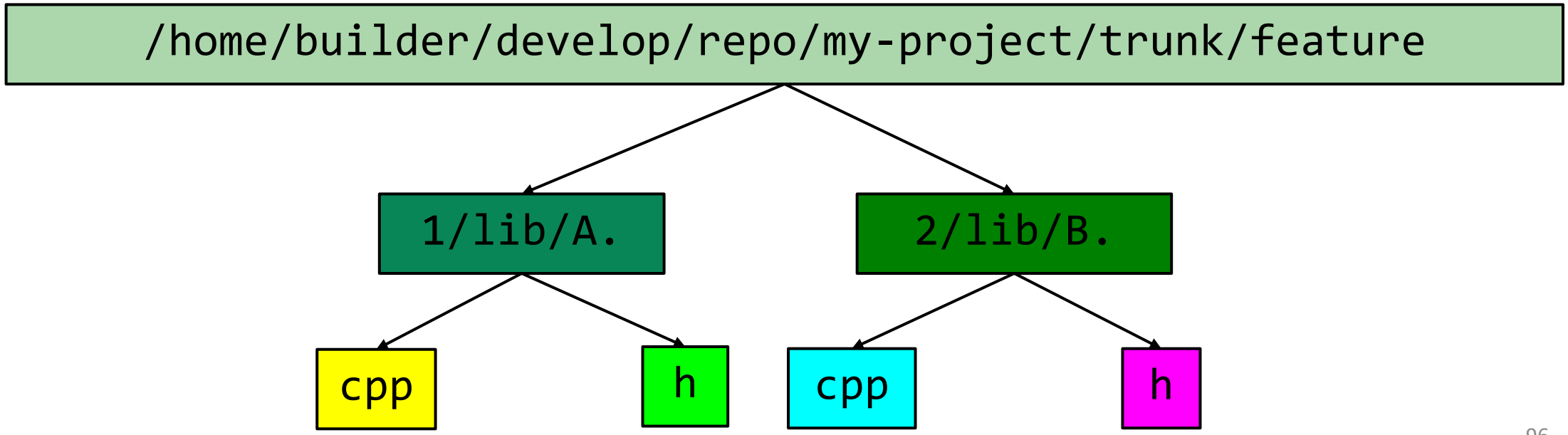


Префиксное дерево

```
/home/builder/develop/repo/my-project/trunk/feature1/lib/A.cpp
/home/builder/develop/repo/my-project/trunk/feature1/lib/A.h
/home/builder/develop/repo/my-project/trunk/feature2/lib/B.cpp
/home/builder/develop/repo/my-project/trunk/feature2/lib/B.h
```

Префиксное дерево

```
/home/builder/develop/repo/my-project/trunk/feature1/lib/A.cpp
/home/builder/develop/repo/my-project/trunk/feature1/lib/A.h
/home/builder/develop/repo/my-project/trunk/feature2/lib/B.cpp
/home/builder/develop/repo/my-project/trunk/feature2/lib/B.h
```



Инкрементальный анализ

Вариант первый

```
int f1(int *ptr);

int main()
{
  return f1(nullptr);
}
```

A.cpp

call site

```
int f2(int *ptr);

int f1(int *ptr)
{
  return f2(ptr);
}
```

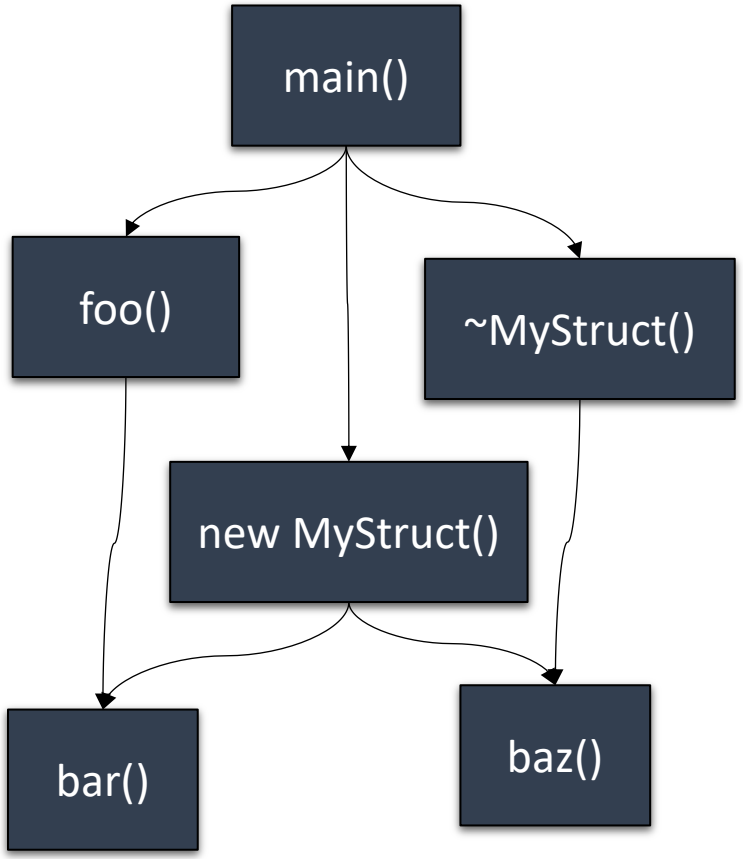
B.cpp

call site

```
int f2(int *ptr)
{
  return *ptr;
}
```

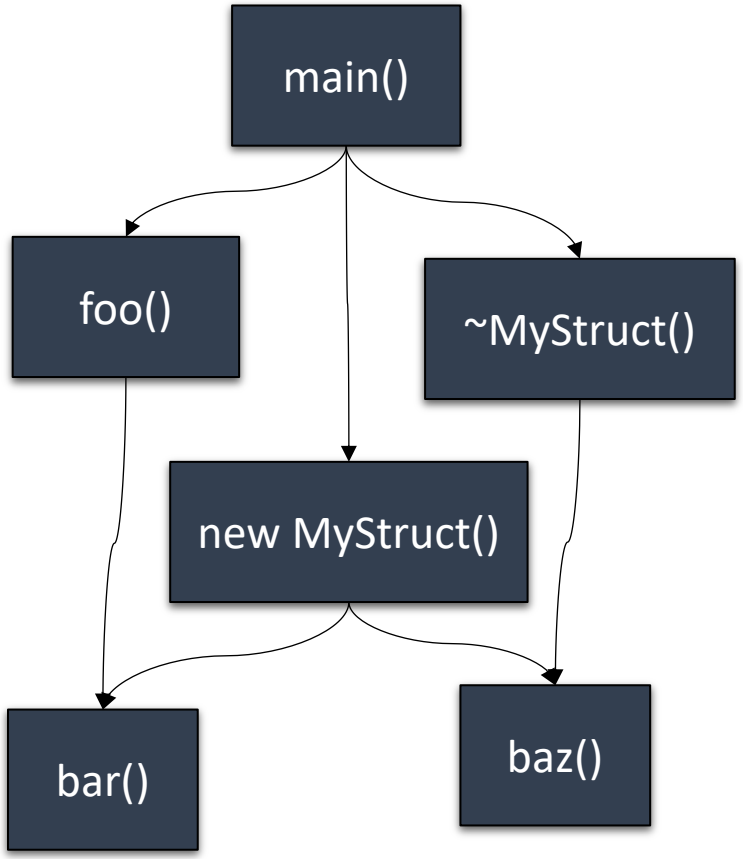
C.cpp

Инкрементальный анализ



call graph, call site

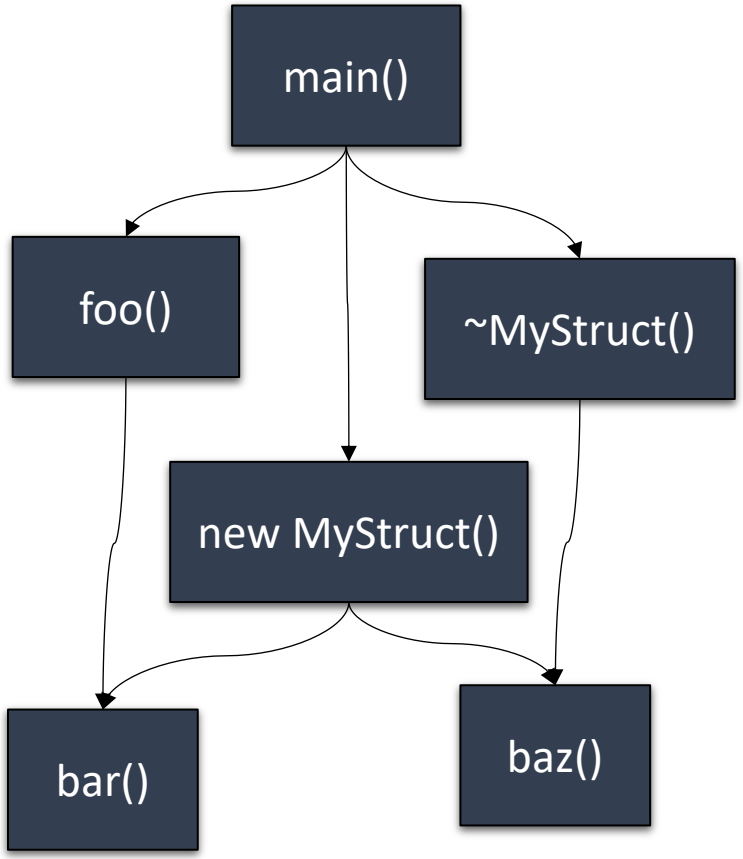
Инкрементальный анализ



call graph, call site

Нужно много раз делать трансляцию кода

Инкрементальный анализ

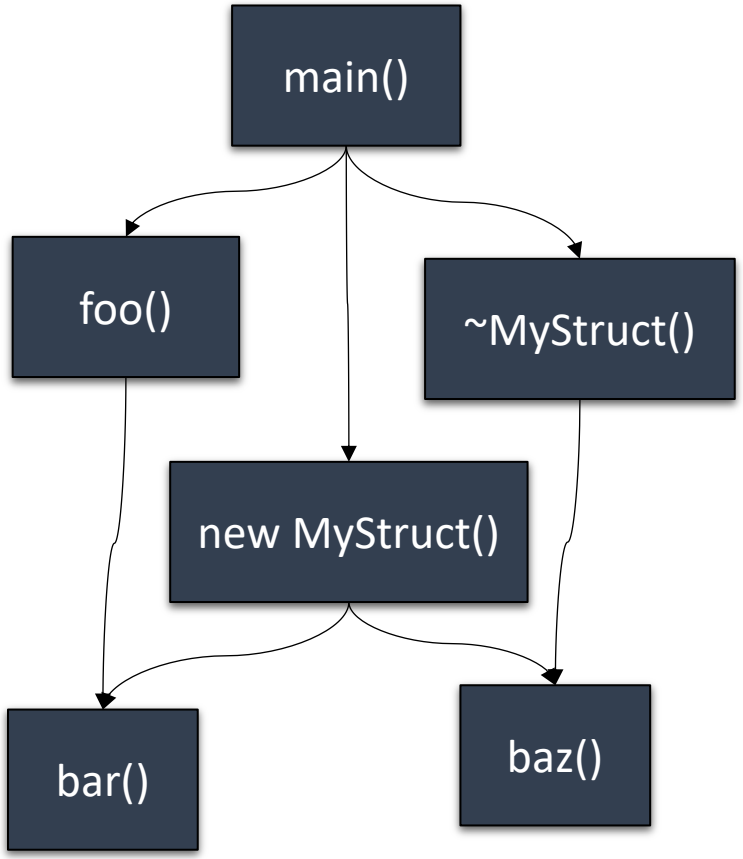


call graph, call site

Нужно много раз делать трансляцию кода

Или нужно промежуточное представление

Инкрементальный анализ



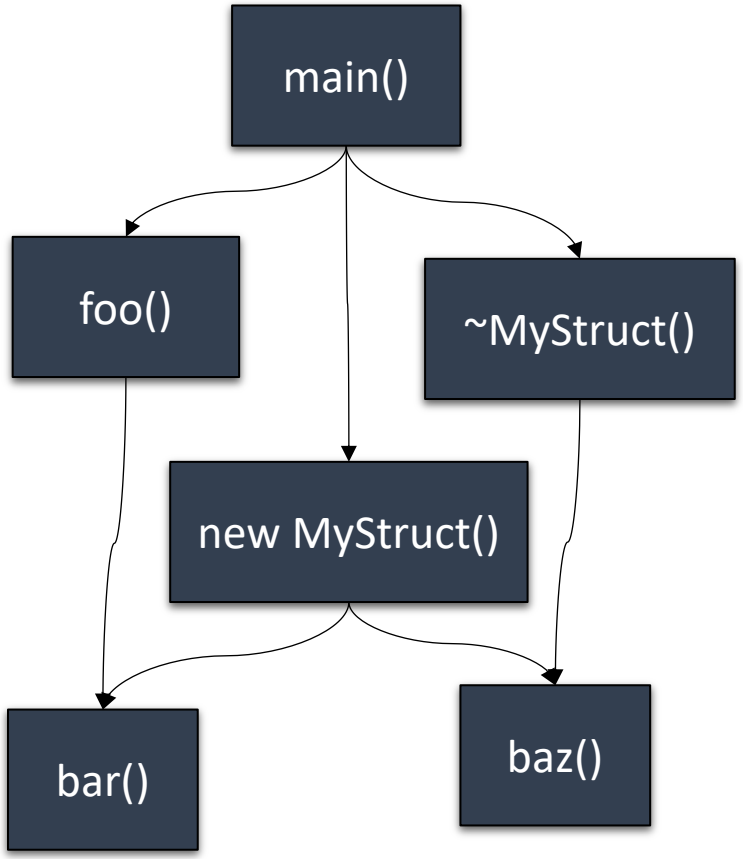
call graph, call site

Нужно много раз делать трансляцию кода

Или нужно промежуточное представление

Проблемы с зацикливанием

Инкрементальный анализ



call graph, call site

Нужно много раз делать трансляцию кода

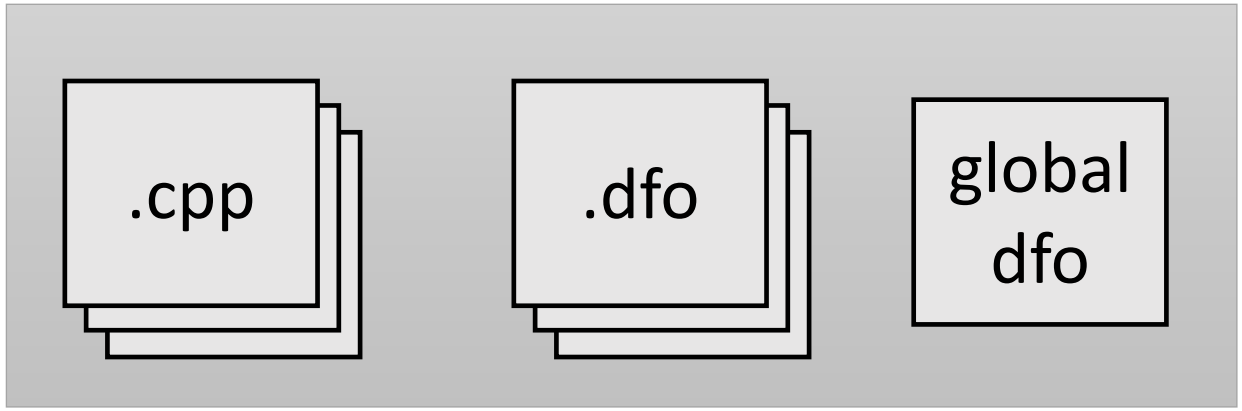
Или нужно промежуточное представление

Проблемы с зацикливанием

Сложно организовать многопоточную обработку

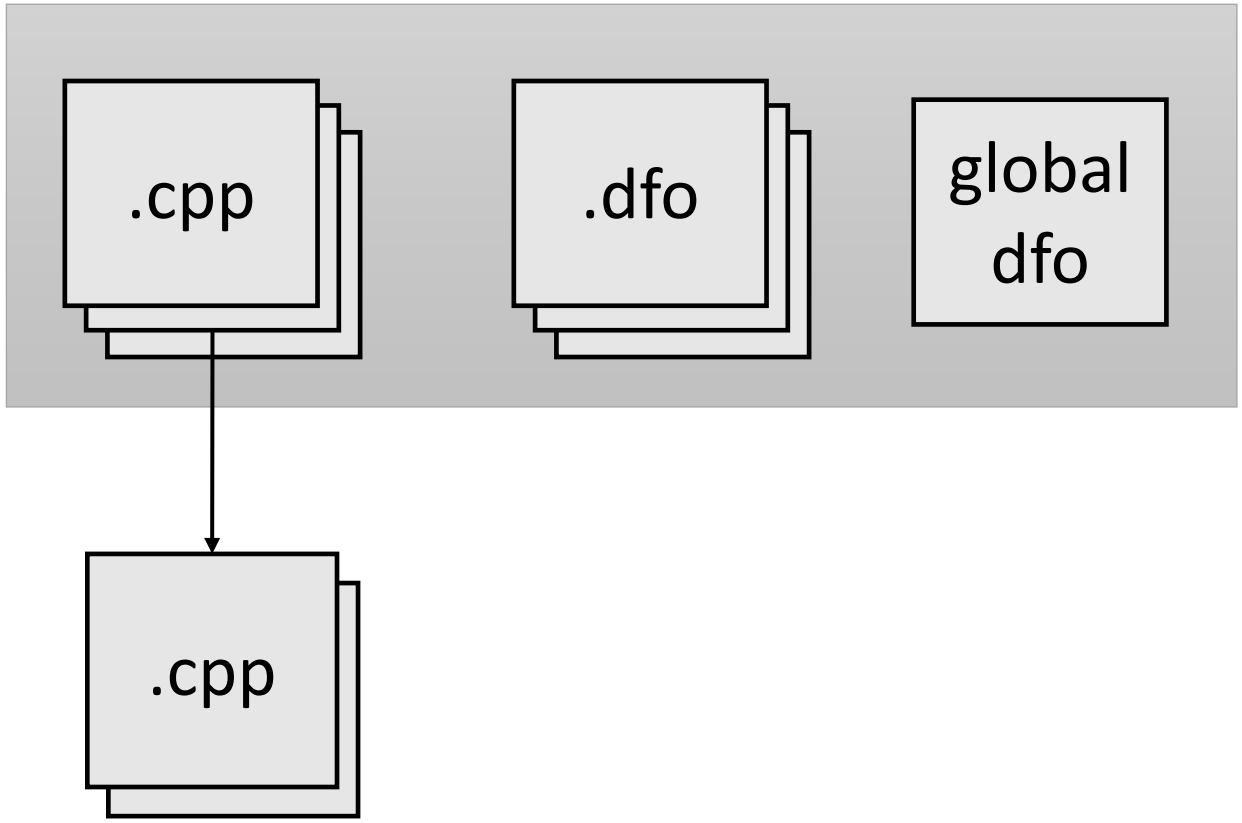
Инкрементальный анализ

Вариант второй



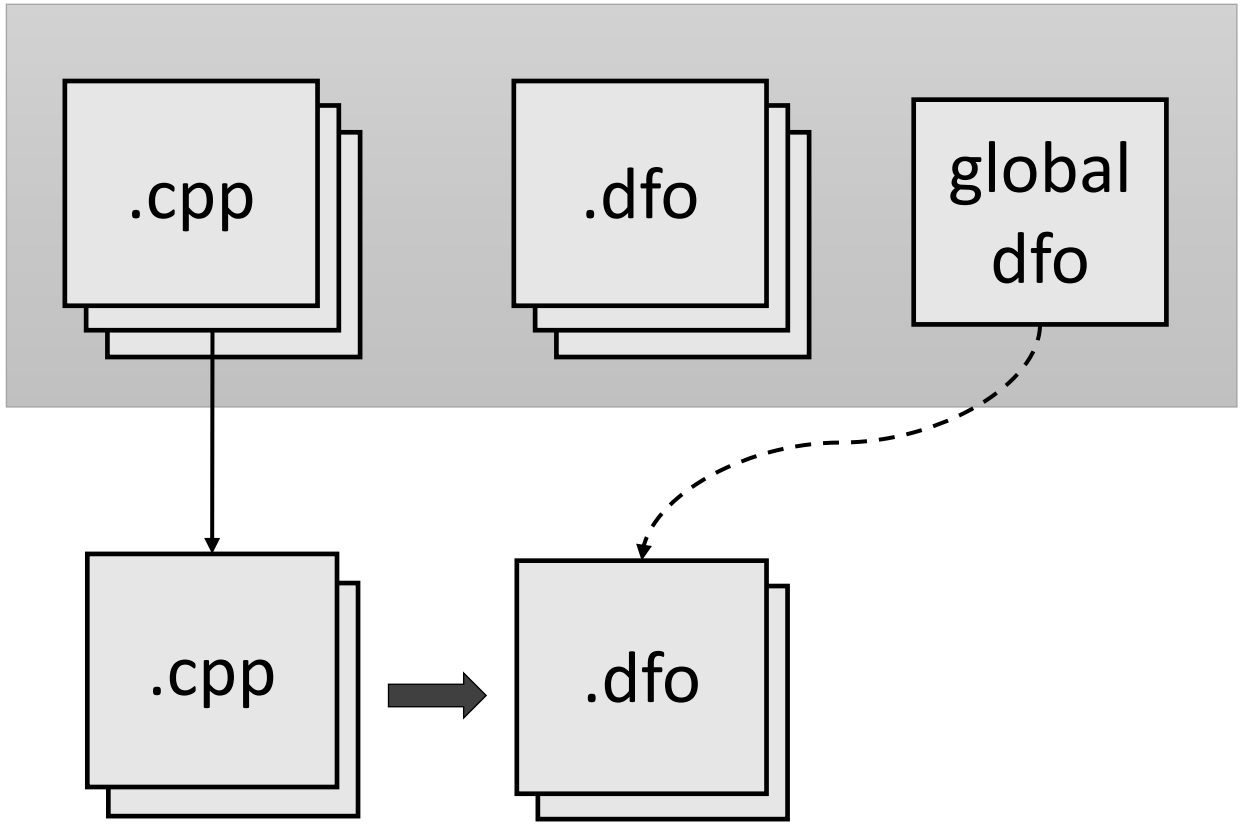
Инкрементальный анализ

Вариант второй



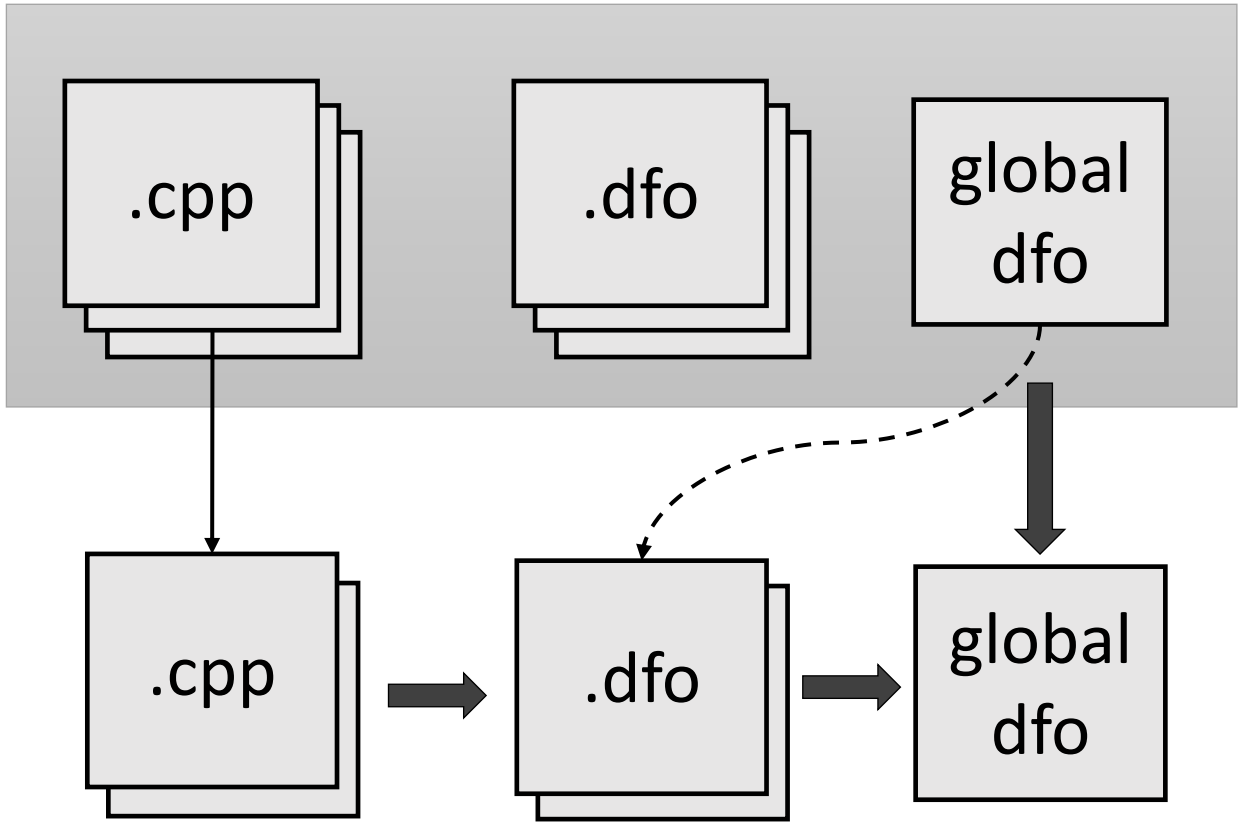
Инкрементальный анализ

Вариант второй



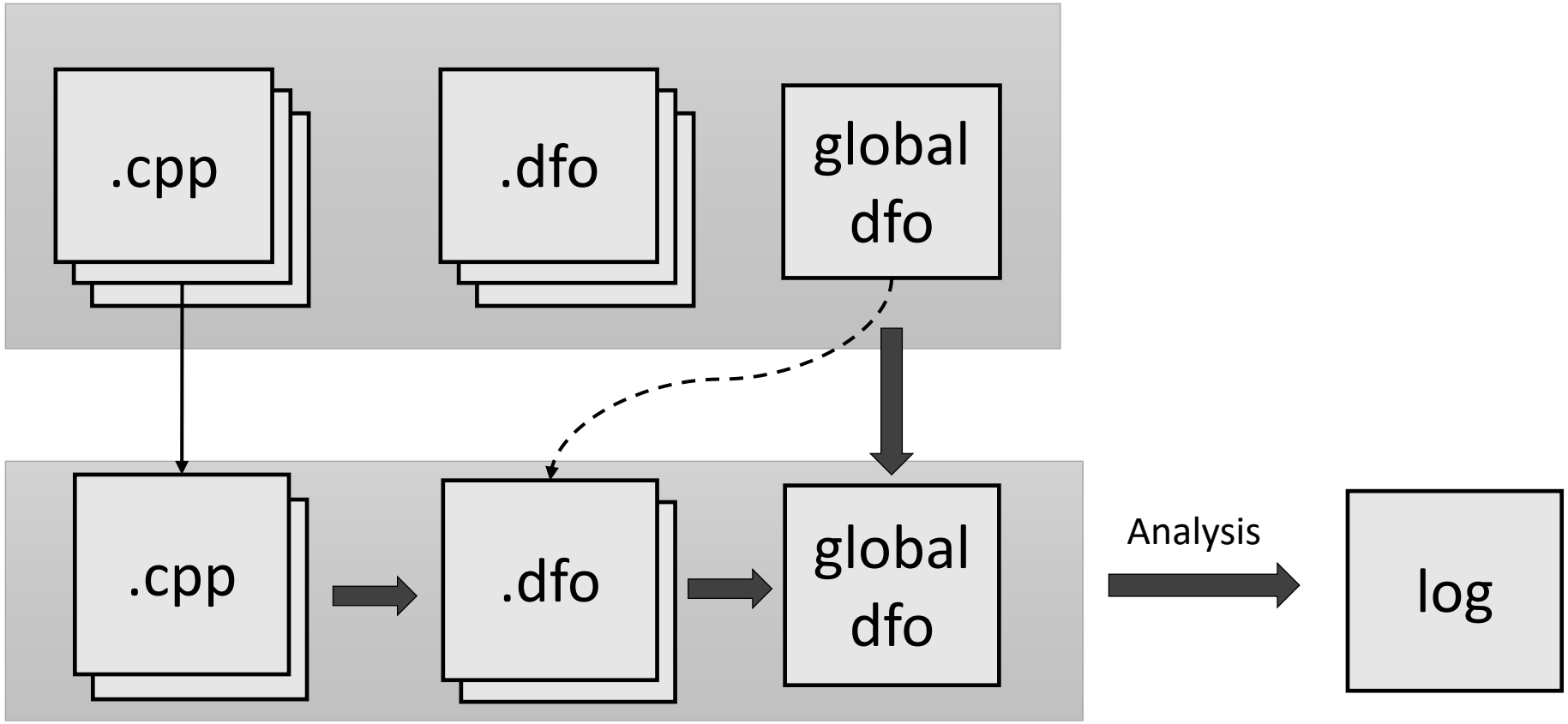
Инкрементальный анализ

Вариант второй



Инкрементальный анализ

Вариант второй



Инкрементальный анализ

Вариант второй

Плюсы:

Инкрементальный анализ

Вариант второй

Плюсы:

- Нет зацикливаний

Инкрементальный анализ

Вариант второй

Плюсы:

- Нет зацикливаний
- Многопоточная обработка

Инкрементальный анализ

Вариант второй

Плюсы:

- Нет зацикливаний
- Многопоточная обработка

Минусы:

Инкрементальный анализ

Вариант второй

Плюсы:

- Нет зацикливаний
- Многопоточная обработка

Минусы:

- Всё еще нужно делать трансляцию изменившихся файлов

Инкрементальный анализ

Вариант второй

Плюсы:

- Нет зацикливаний
- Многопоточная обработка

Минусы:

- Всё еще нужно делать трансляцию изменившихся файлов
- Глубина анализа не учитывает вызовы функций

Посмотрим, что удалось найти

Midnight Commander

```
// editcmd.c
gboolean
edit_close_cmd (WEdit * edit)
{
    // ....
    Widget *w = WIDGET (edit);
    WGroup *g = w->owner;
    if (edit->locked != 0)
        unlock_file(edit->filename_vpath);
    group_remove_widget (w);
    widget_destroy (w); // <=
    // ....
}
```

Midnight Commander

```
// widget-common.c
void widget_destroy (Widget * w)
{
    send_message (w, NULL, MSG_DESTROY, 0, NULL);
    g_free (w);
}
```

Midnight Commander

```
// editcmd.c
gboolean
edit_close_cmd (WEdit * edit)
{
    // ....
    widget_destroy (w);
    if (edit_widget_is_editor(CONST_WIDGET (g->current->data)))
        edit = (WEdit *) (g->current->data);
    else
    {
        edit = find_editor(DIALOG (g));
        if (edit != NULL)
            widget_select (w); // <=
    }
}
```

Midnight Commander

```
// widget-common.c
void widget_select (Widget * w)
{
    WGroup *g;
    if (!widget_get_options (w, WOP_SELECTABLE))
        return;
    // ....
}
```

Midnight Commander

```
// widget-common.h
static inline gboolean
widget_get_options (const Widget * w, widget_options_t options)
{
    return ((w->options & options) == options);
}
```

Pull request с исправлением ошибки уже принят разработчиками

Codelite

```
// args.c
extern void eFree (void *const ptr);

extern void argDelete (Arguments* const current)
{
  Assert (current != NULL);
  if (current->type ==  ARG_STRING  &&  current->item != NULL)
    eFree (current->item);
  memset (current, 0, sizeof (Arguments));  // <=
  eFree (current);                          // <=
}
```

Вызов *memset* может быть удален при использовании оптимизаций LTO

Codelite

```
// routines.c
extern void eFree (void *const ptr)
{
  Assert (ptr != NULL);
  free (ptr); // <=
}
```

Всё!