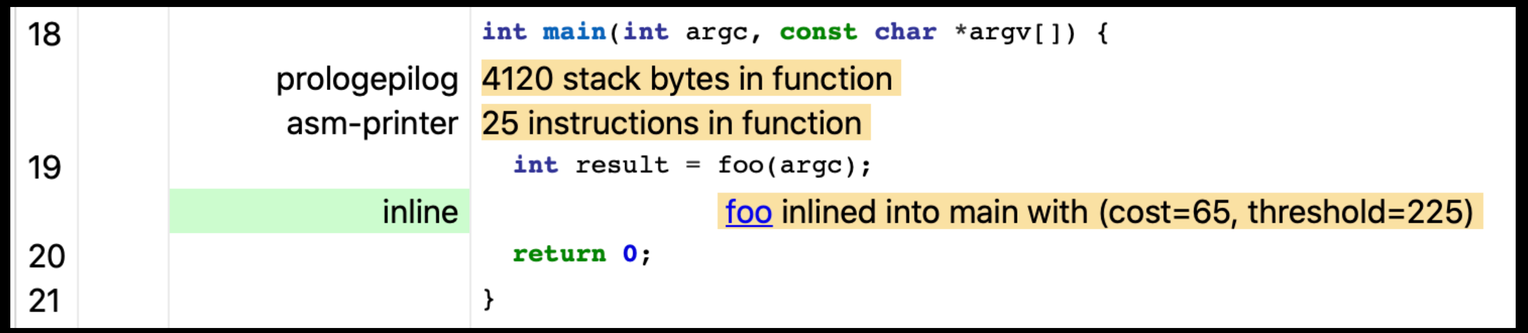
Optimization Remarks Update

Francis Visoiu Mistrih

Optimization Remarks



opt-viewer.py

Challenges

- YAML doesn’t scale well
- Discovering remarks
- Processing remarks

clang / RelWithDebInfo / macOS

- Significant size impact

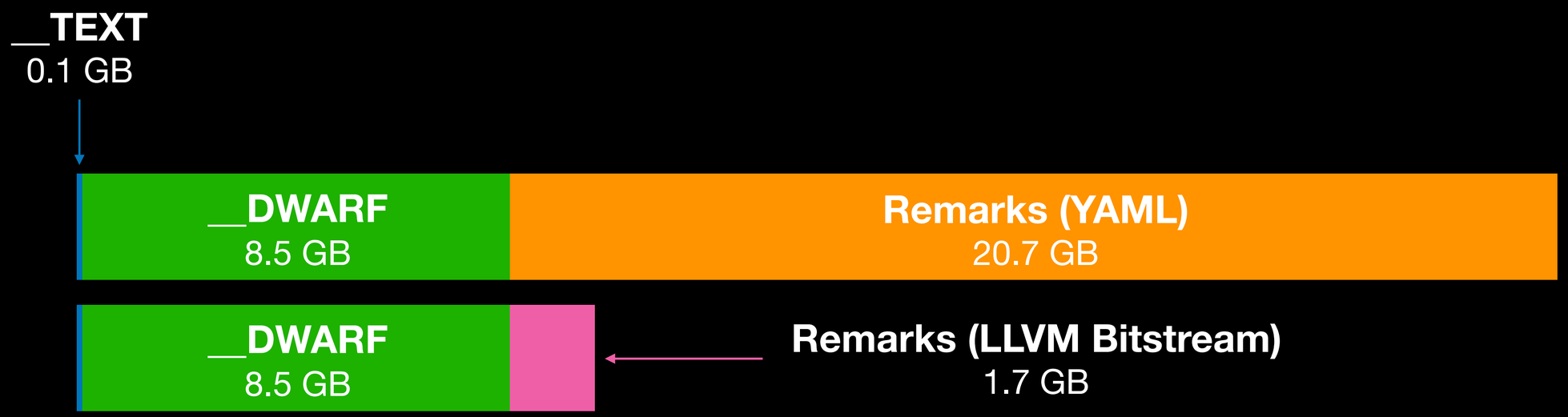
clang / RelWithDebInfo / macOS

- Significant size impact
- Slow to parse: **~4 minutes** for 27 million remarks

clang / RelWithDebInfo / macOS

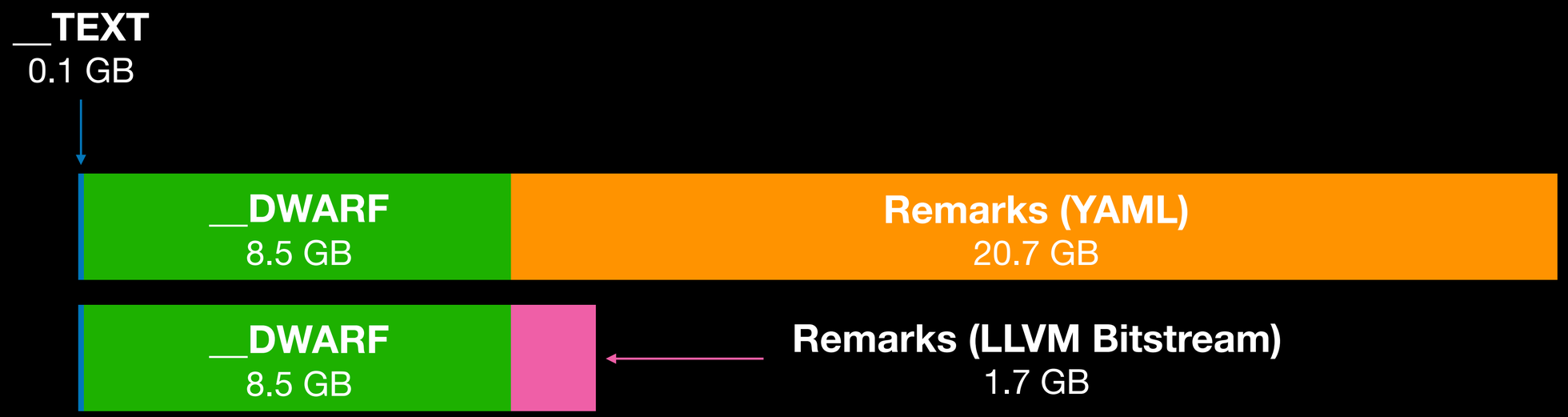
- Significant size impact
- Slow to parse: **~4 minutes** for 27 million remarks
- Compile-time impact: **5% slowdown**

clang / RelWithDebInfo / macOS



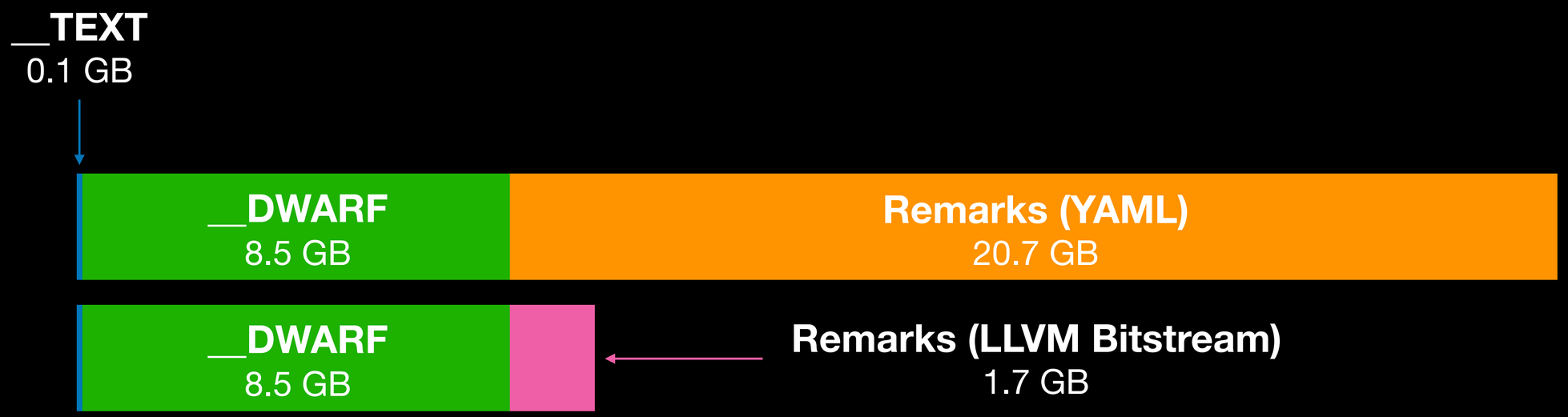
- Significant size impact
- Slow to parse: **~4 minutes** for 27 million remarks
- Compile-time impact: **5% slowdown**

clang / RelWithDebInfo / macOS



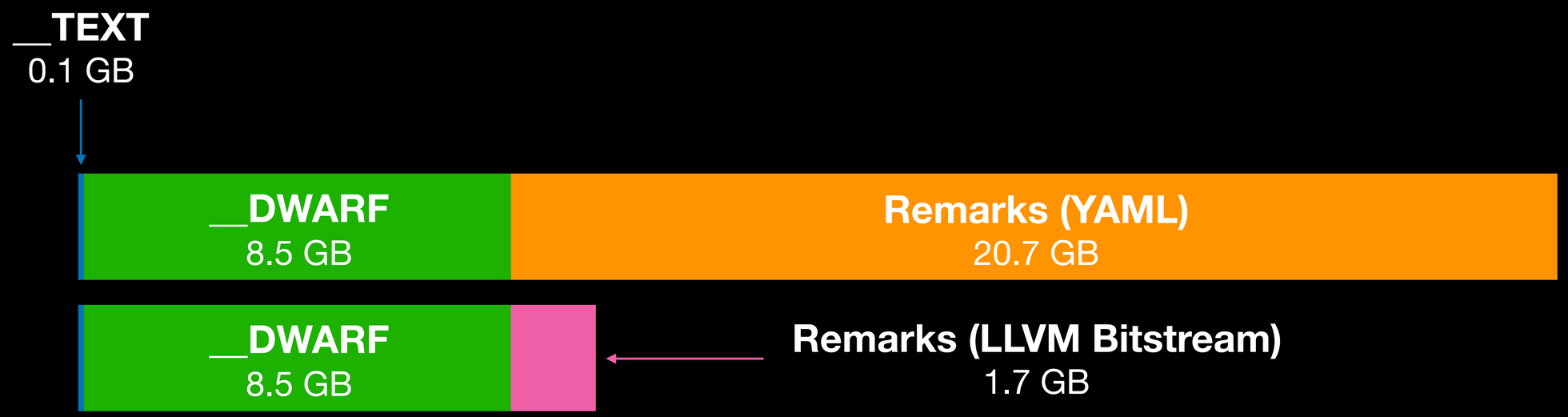
- ~~Significant size impact~~ **12x smaller**
- Slow to parse: **~4 minutes** for 27 million remarks
- Compile-time impact: **5% slowdown**

clang / RelWithDebInfo / macOS



- ~~Significant size impact~~ **12x smaller**
- ~~Slow to parse: **~4 minutes**~~ **27 seconds (8x faster)**
- Compile-time impact: **5% slowdown**

clang / RelWithDebInfo / macOS



- ~~Significant size impact~~ 12x smaller
- ~~Slow to parse: ~4 minutes~~ 27 seconds (8x faster)
- ~~Compile-time impact: 5% slowdown~~ 2% slowdown

clang / RelWithDebInfo / macOS

Extra Benefits

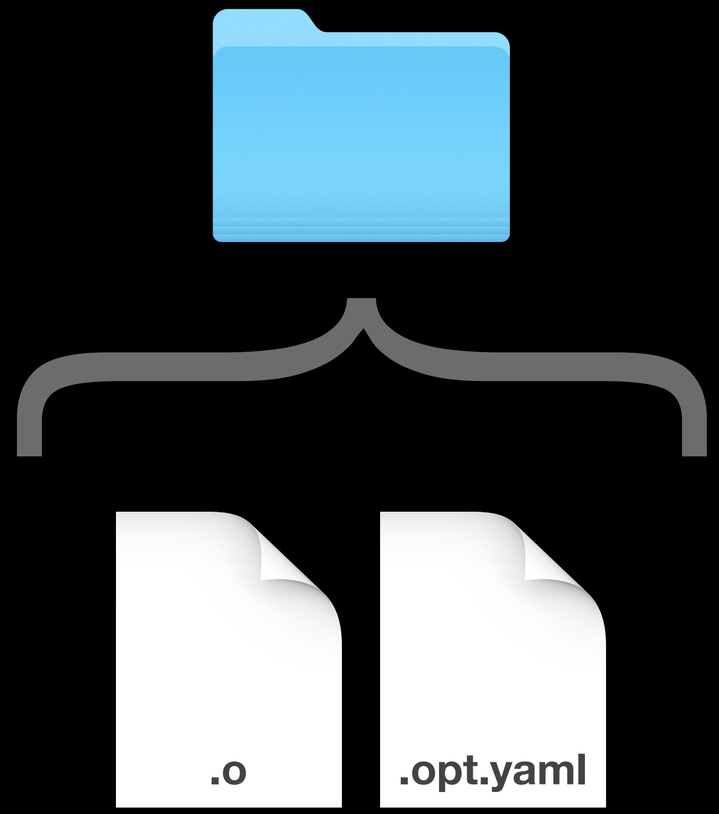
- Well-known format in the community
- Testing and tooling like `llvm-bcanalyzer`
- Versioning
- Flexible metadata

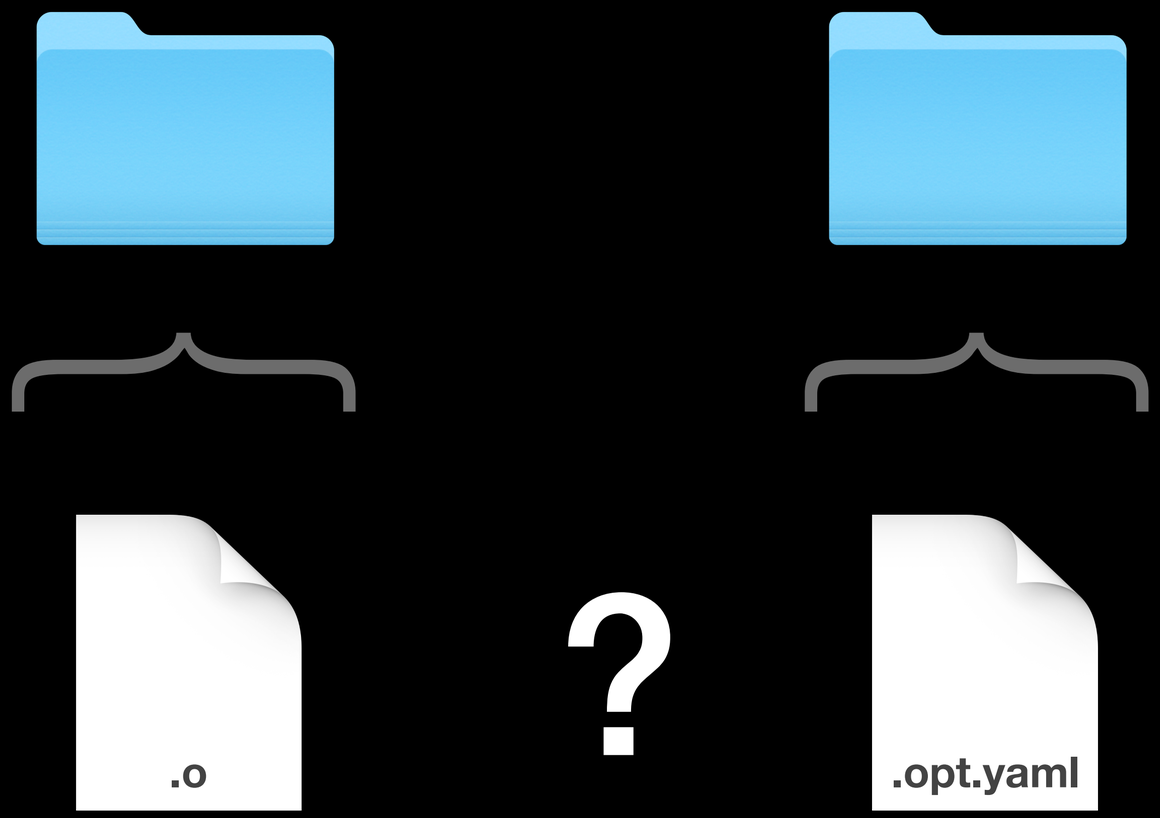
libRemarks

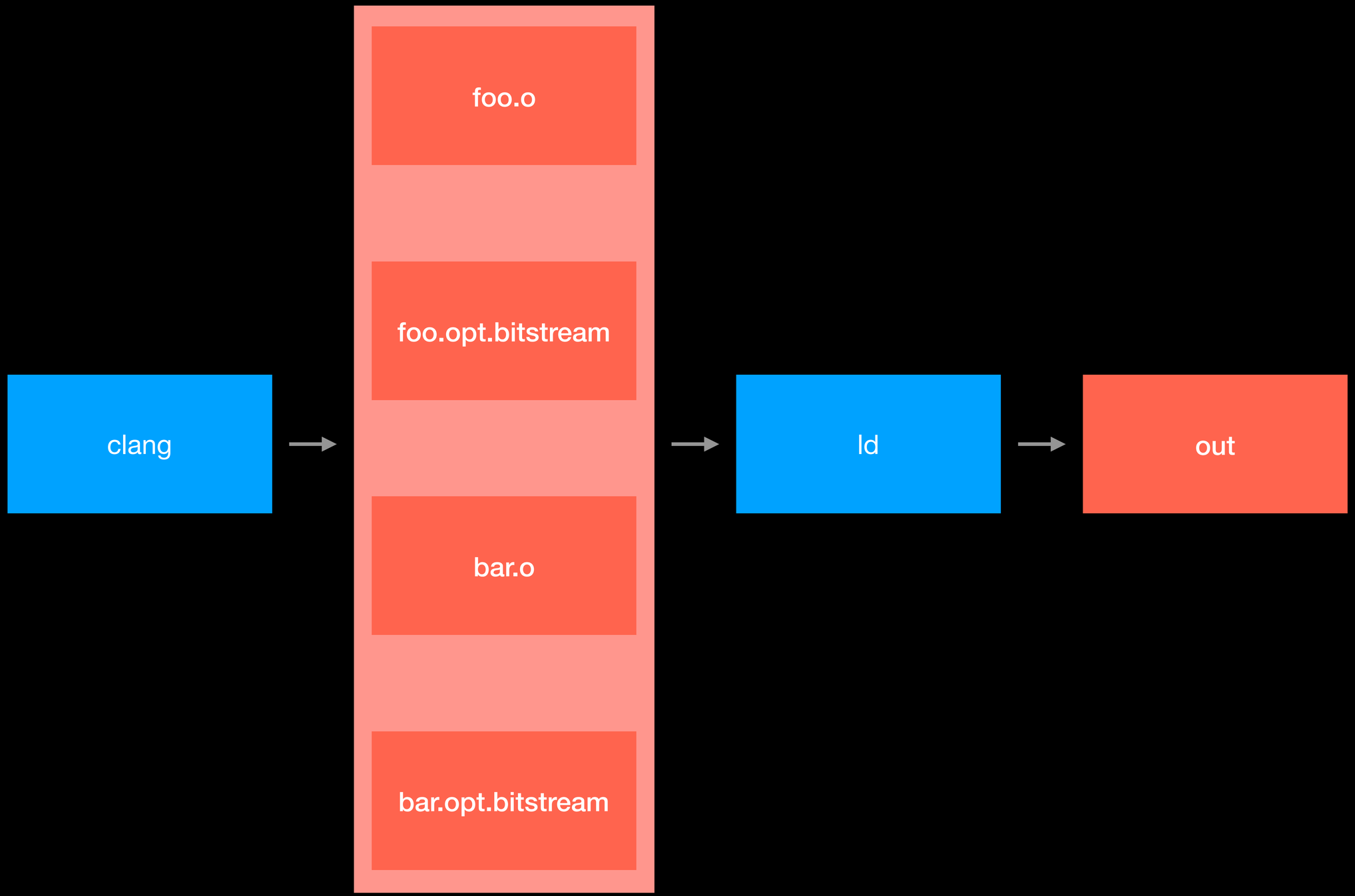
```c
#include "llvm-c/Remarks.h"

LLVMRemarkParserRef Parser = LLVMRemarkParserCreateYAML(Buf, Size);
LLVMRemarkEntryRef Remark = NULL;
while ((Remark = LLVMRemarkParserGetNext(Parser))) {
  // use Remark
}
```

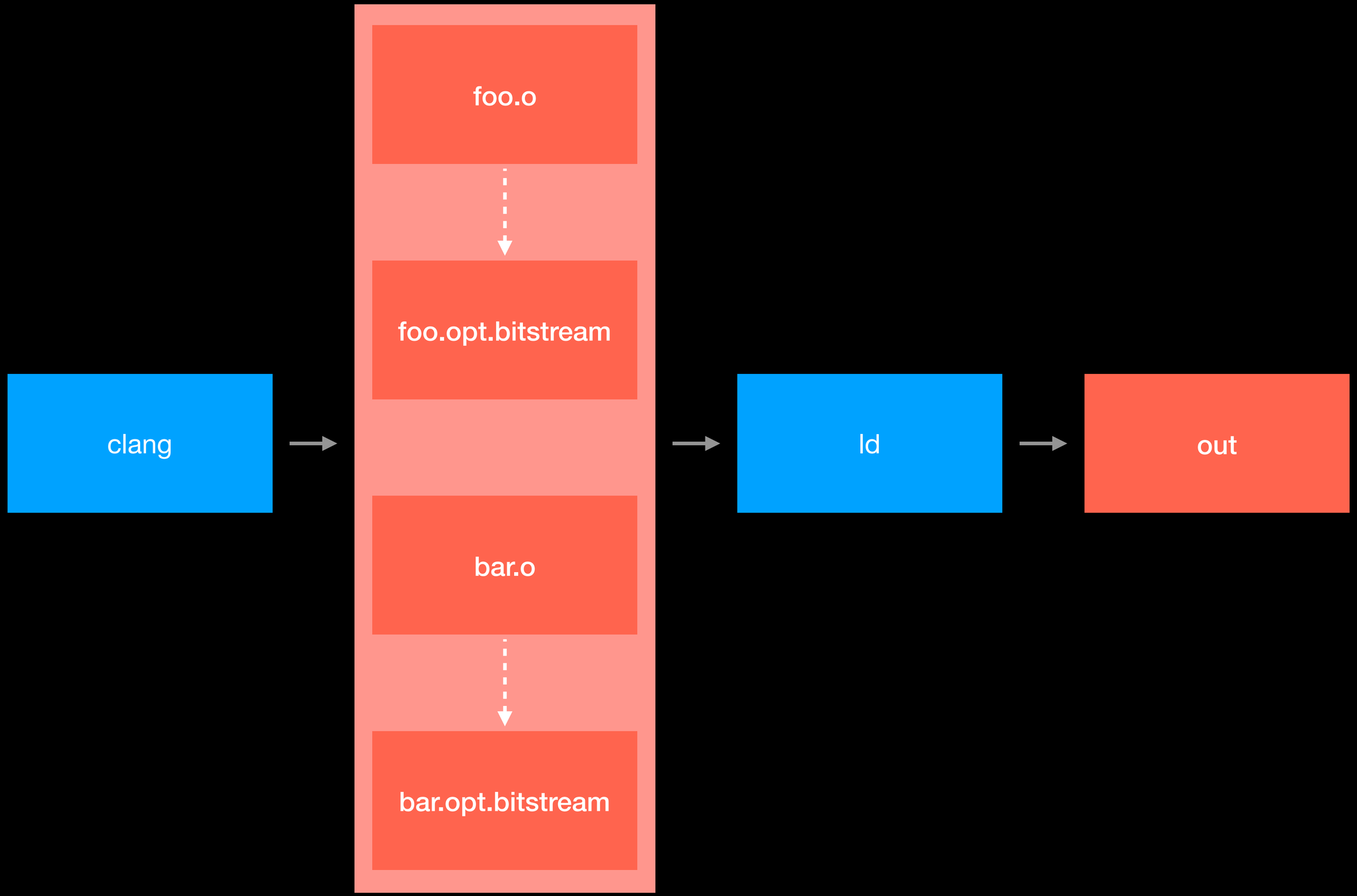
libRemarks

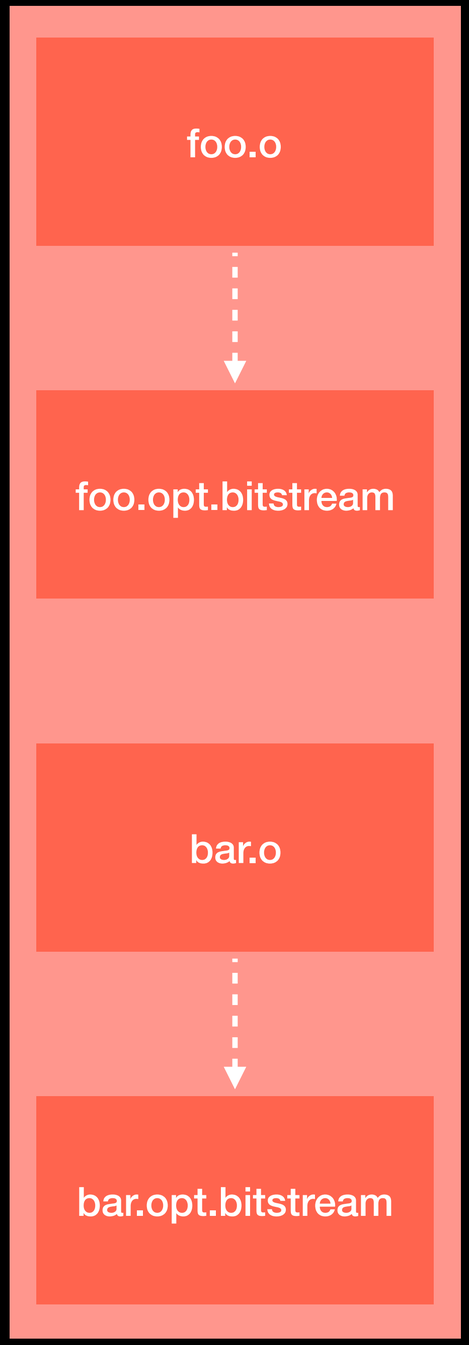
```c
#include "llvm-c/Remarks.h"

LLVMRemarkParserRef Parser = LLVMRemarkParserCreateBitstream(Buf, Size);
LLVMRemarkEntryRef Remark = NULL;
while ((Remark = LLVMRemarkParserGetNext(Parser))) {
  // use Remark
}
```


.o
.opt.yaml


.o
?
.opt.yaml


clang
foo.o
foo.opt.bitstream
bar.o
bar.opt.bitstream
ld
out


foo.o
foo.opt.bitstream
clang
ld
out
bar.o
bar.opt.bitstream


foo.o
foo.opt.bitstream
bar.o
bar.opt.bitstream


foo.o
__remarks
52 4d 52 4b 01
08 00 00 17 00
00 00 07 01 b2
40 b4 42 39 d0
43 38 3c 3c c1
28 bc 83 3b d0
43 38 a4 83 3b
94 83 3c 80 41
3a b8 83 39 bc
c3 41 80 38 06
08 14 22 1e 9a
61 16 e8 41 1e
d2 c1 1d ce 01
0c e8 21 1c c4
81 1d ca 41 71
40 1f 1c a2 14
f0 81 1e ca 41
1e dc 21 1c d8
foo.opt.bitstream

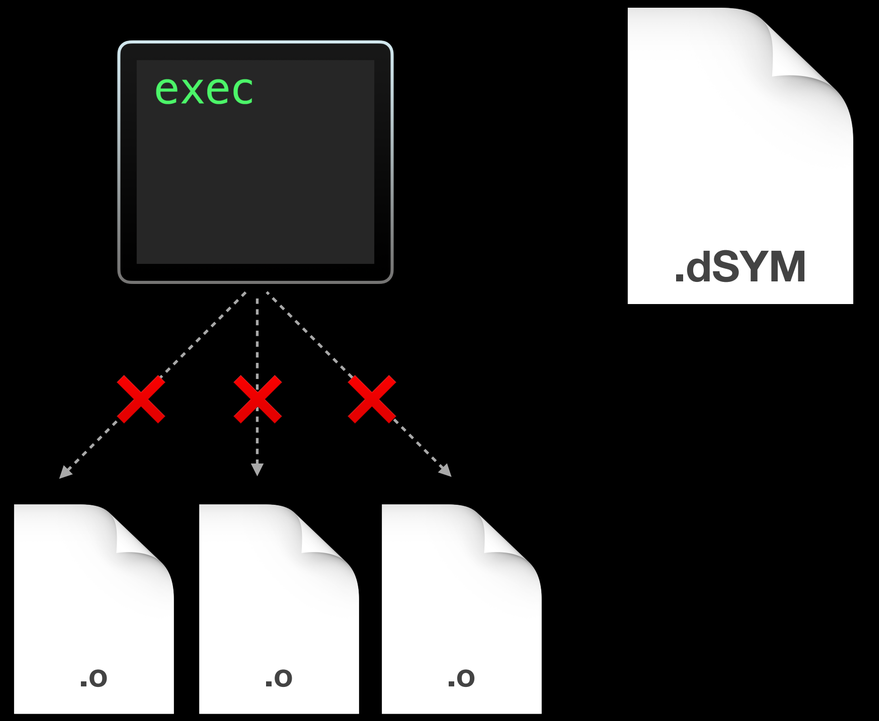

foo.o
__remarks
52 4d 52 4b 01
08 00 00 17 00
00 00 07 01 b2
40 b4 42 39 d0
43 38 3c 3c c1
28 bc 83 3b d0
43 38 a4 83 3b
94 83 3c 80 41
<file path>
d2 c1 1d ce 01
0c e8 21 1c c4
81 1d ca 41 71
40 1f 1c a2 14
f0 81 1e ca 41
1e dc 21 1c d8
foo.opt.bitstream

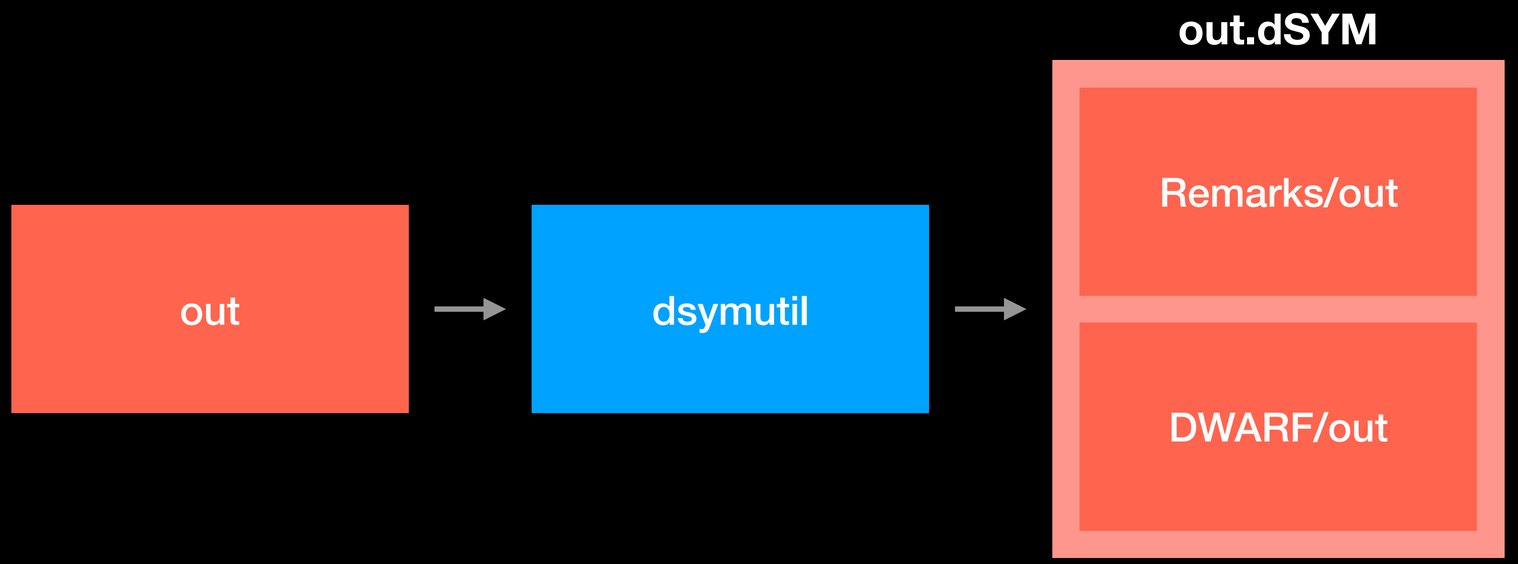
Remarks Section

- Only in object files
- Not included in the linked binary
- Accessible through the Debug Map
- clang’s remarks section: 0.27GB


exec


.dSYM


exec
.dSYM
.o
.o
.o


out.dSYM
out
dsymutil
Remarks/out
DWARF/out

Get Involved

- Python bindings for integration with `opt-viewer.py`
- llvm-remarkutil: convert, merge, extract, etc.
- Faster lookup: build an index
- Remark classes: verbosity, accuracy, etc.
- Tablegen: better remark structure
- Per-remark documentation: possible actions, reasons, etc.

Thanks!