Kompilieren und Linken

Proseminar „Effiziente Programmierung in C“

Joshua Lowe

6. Juni 2021

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
I use arch btw

Universität Hamburg

Gliederung

Compiler

Linker

GCC

Compiler

Struktur

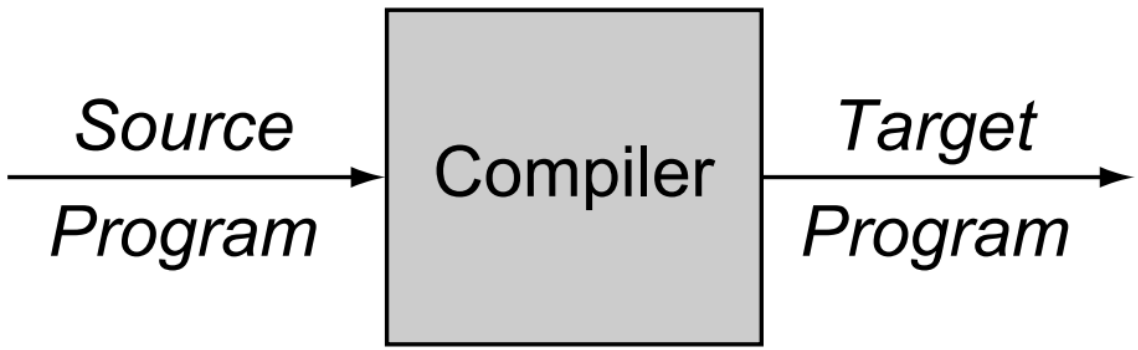


Abbildung: Compiler [5]

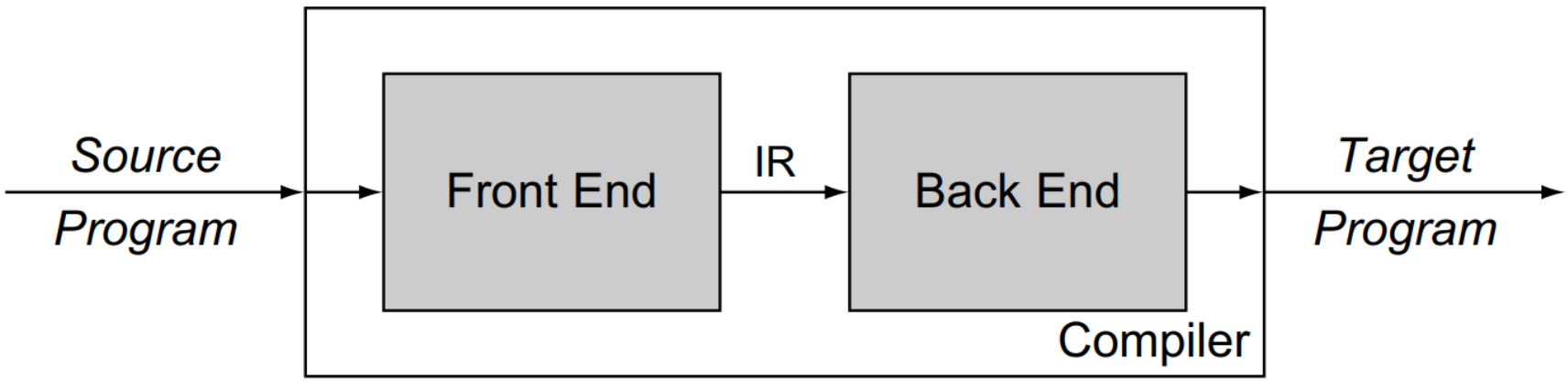


Abbildung: Struktur [5]

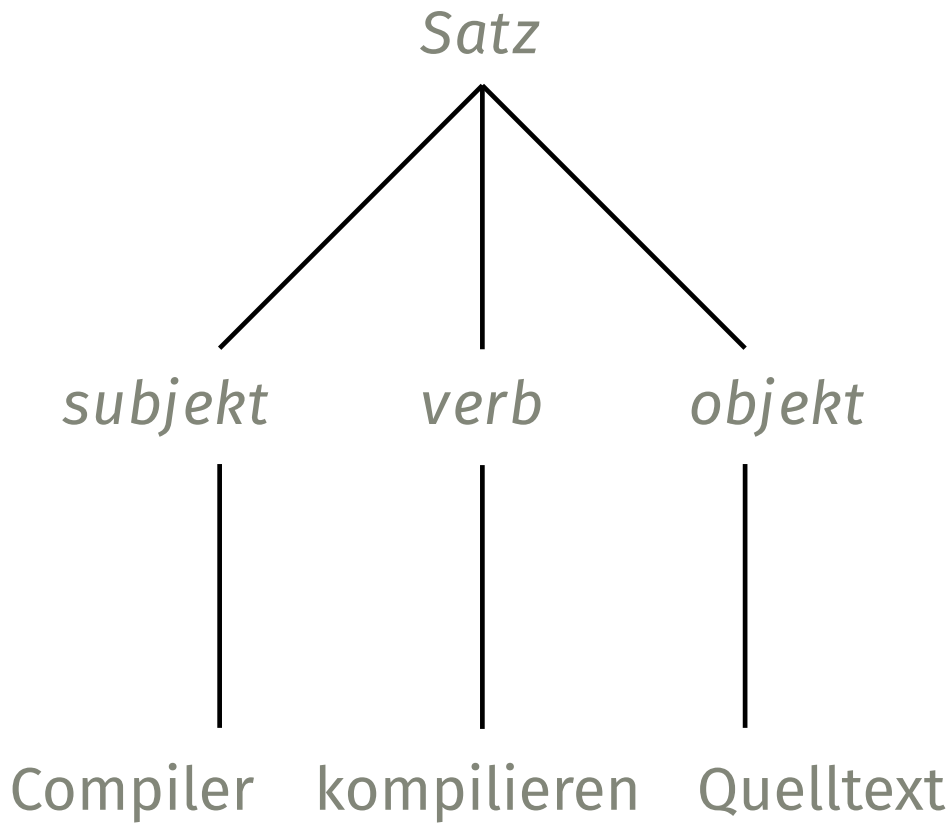
Analysephase

- ⟨Token Name, Attribute⟩
- ⟨bz, ”summe”⟩ ⟨=⟩ ⟨num, 9⟩ ⟨+⟩ ⟨num, 10⟩ ⟨;⟩

```
1  /* void :) */
2  void main() {
3    /* addition much wow */
4    int summe;
5    summe = 9 + 10;
6
7    while (1) {
8      break;
9    }
10 }
```

$$
\begin{aligned}
\textit{Satz} &\rightarrow \textit{subjekt verb objekt} \\
\textit{subjekt} &\rightarrow \text{Compiler} \mid \text{Linker} \mid \text{Computer} \\
\textit{verb} &\rightarrow \text{kompilieren} \mid \text{schreiben} \mid \text{malen} \\
\textit{objekt} &\rightarrow \text{Quelltext} \mid \text{Musik} \mid \text{Bilder}
\end{aligned}
$$

Compiler kompilieren Quelltext


Satz
subjekt
verb
objekt
Compiler
kompilieren
Quelltext

Synthesephase

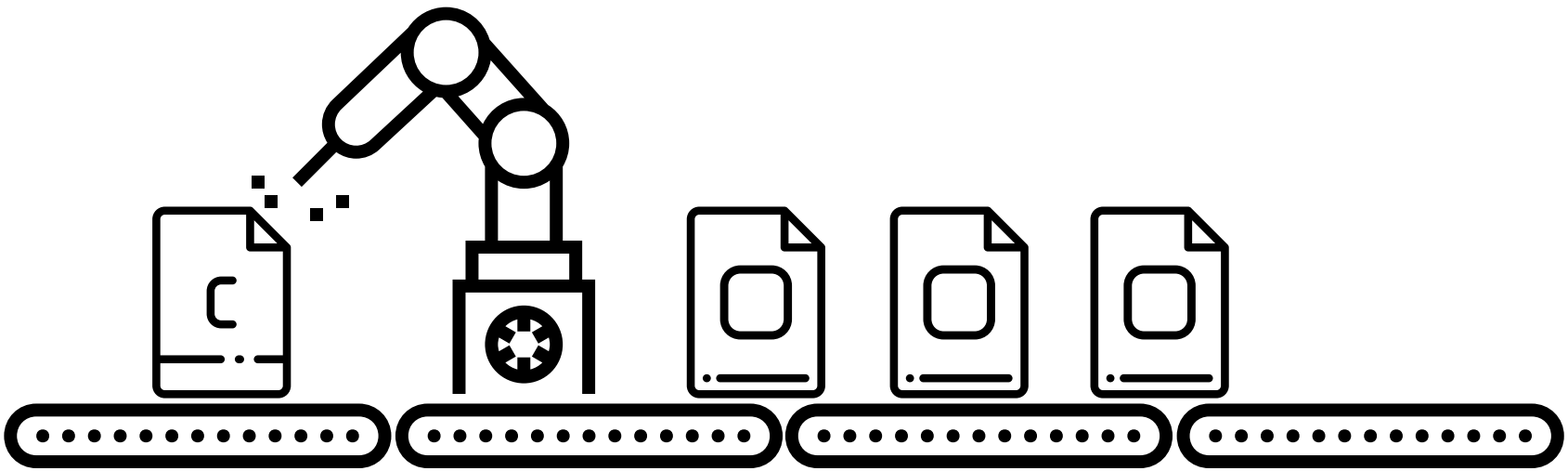
Start	Operations
1	loadAI r_{arp},@a ⇒ r_1
4	add r_1,r_1 ⇒ r_1
5	loadAI r_{arp},@b ⇒ r_2
8	mult r_1,r_2 ⇒ r_1
10	loadAI r_{arp},@c ⇒ r_2
13	mult r_1,r_2 ⇒ r_1
15	loadAI r_{arp},@d ⇒ r_2
18	mult r_1,r_2 ⇒ r_1
20	storeAI r_1 ⇒ r_{arp},@a

(a) Original Code

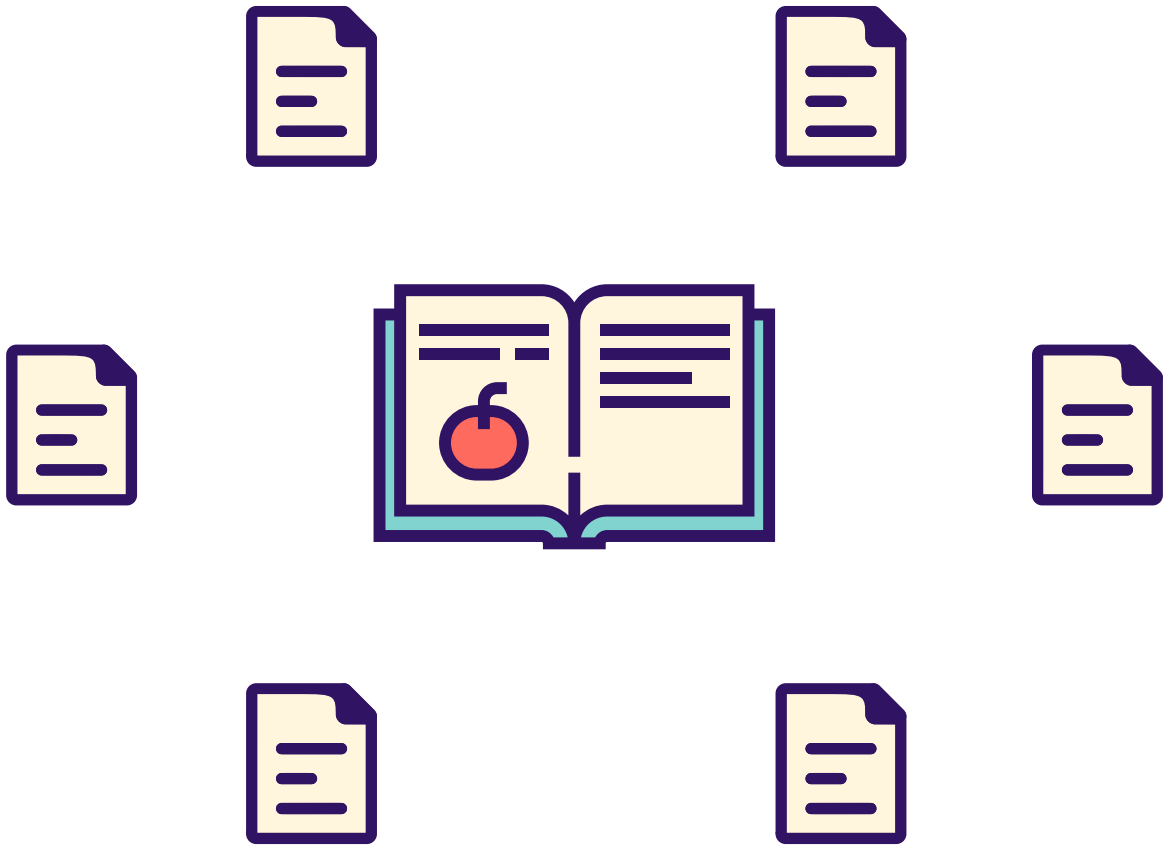
Start	Operations
1	loadAI r_{arp},@a ⇒ r_1
2	loadAI r_{arp},@b ⇒ r_2
3	loadAI r_{arp},@c ⇒ r_3
4	add r_1,r_1 ⇒ r_1
5	mult r_1,r_2 ⇒ r_1
6	loadAI r_{arp},@d ⇒ r_2
7	mult r_1,r_3 ⇒ r_1
9	mult r_1,r_2 ⇒ r_1
11	storeAI r_1 ⇒ r_{arp},@a

(b) Scheduled Code

Abbildung: Scheduling [5]


C

Linker



Objekt Datei

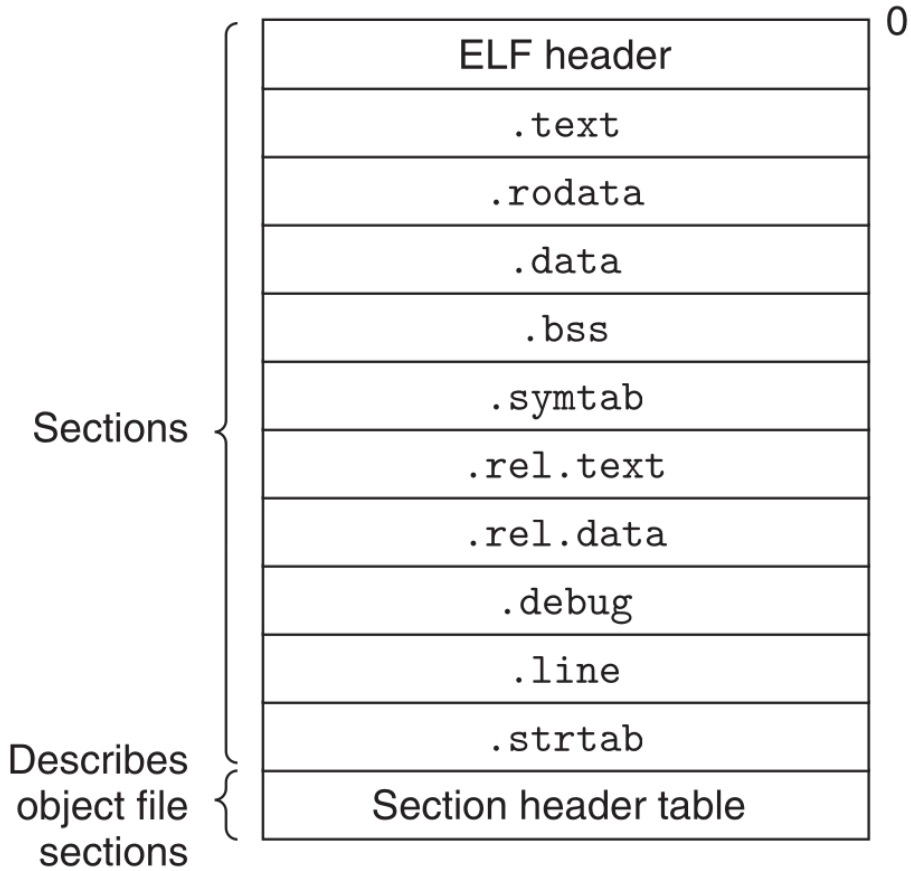


Abbildung: Objekt Datei [3]

Linken

```c
/* main.c */
int sum(int a, int b);
static int sub(int a, int b) { return a - b; }

int main() {
  int differenz = sub(42, 0);
  int summe = sum(22, 20);
  return 0;
}
```

```c
/* sum.c */
int sum(int a, int b) { return a + b; }
```

```
0000000000000012 <main>:
  12: 55                    push   %rbp
  13: 48 89 e5              mov    %rsp,%rbp
  16: 48 83 ec 10           sub    $0x10,%rsp
  1a: be 00 00 00 00        mov    $0x0,%esi
  1f: bf 2a 00 00 00        mov    $0x2a,%edi
  24: e8 d7 ff ff ff        call   0 <sub>
  29: 89 45 f8              mov    %eax,-0x8(%rbp)
  2c: be 14 00 00 00        mov    $0x14,%esi
  31: bf 16 00 00 00        mov    $0x16,%edi
  36: e8 00 00 00 00        call   3b <main+0x29>
      37: R_X86_64_PLT32  sum-0x4
  3b: 89 45 fc              mov    %eax,-0x4(%rbp)
  3e: b8 00 00 00 00        mov    $0x0,%eax
  43: c9                    leave
  44: c3                    ret
```

objdump -dx main.o

```
3   Offset          Info           Type           Sym. Value    Sym. Name + Addend
4 000000000037  000600000004 R_X86_64_PLT32    0000000000000000 sum - 4
```

readelf -r main.o

```
000000000000112b <main>:
    112b: 55                    push   %rbp
    112c: 48 89 e5              mov    %rsp,%rbp
    112f: 48 83 ec 10           sub    $0x10,%rsp
    1133: be 00 00 00 00        mov    $0x0,%esi
    1138: bf 2a 00 00 00        mov    $0x2a,%edi
    113d: e8 d7 ff ff ff        call   1119 <sub>
    1142: 89 45 f8              mov    %eax,-0x8(%rbp)
    1145: be 14 00 00 00        mov    $0x14,%esi
    114a: bf 16 00 00 00        mov    $0x16,%edi
    114f: e8 0a 00 00 00        call   115e <sum>
    1154: 89 45 fc              mov    %eax,-0x4(%rbp)
    1157: b8 00 00 00 00        mov    $0x0,%eax
    115c: c9                    leave
    115d: c3                    ret

000000000000115e <sum>:
    115e: 55                    push   %rbp
    115f: 48 89 e5              mov    %rsp,%rbp
    1162: 89 7d fc              mov    %edi,-0x4(%rbp)
```

objdump -dx a.out

Linken mit Bibliotheken

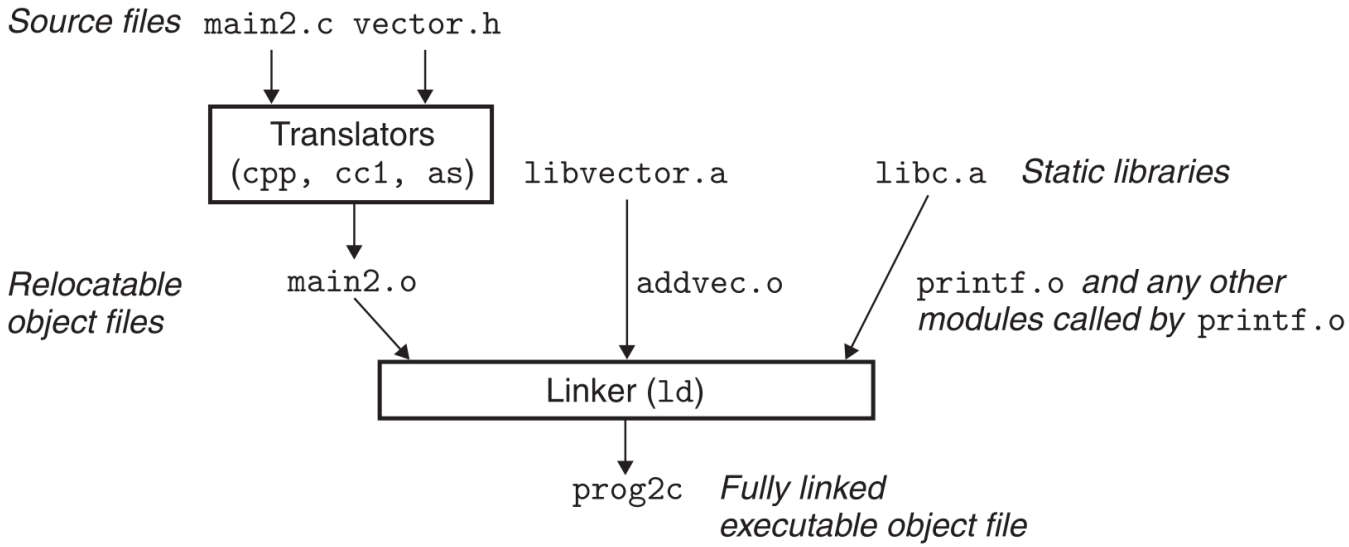


Abbildung: static linking [3]

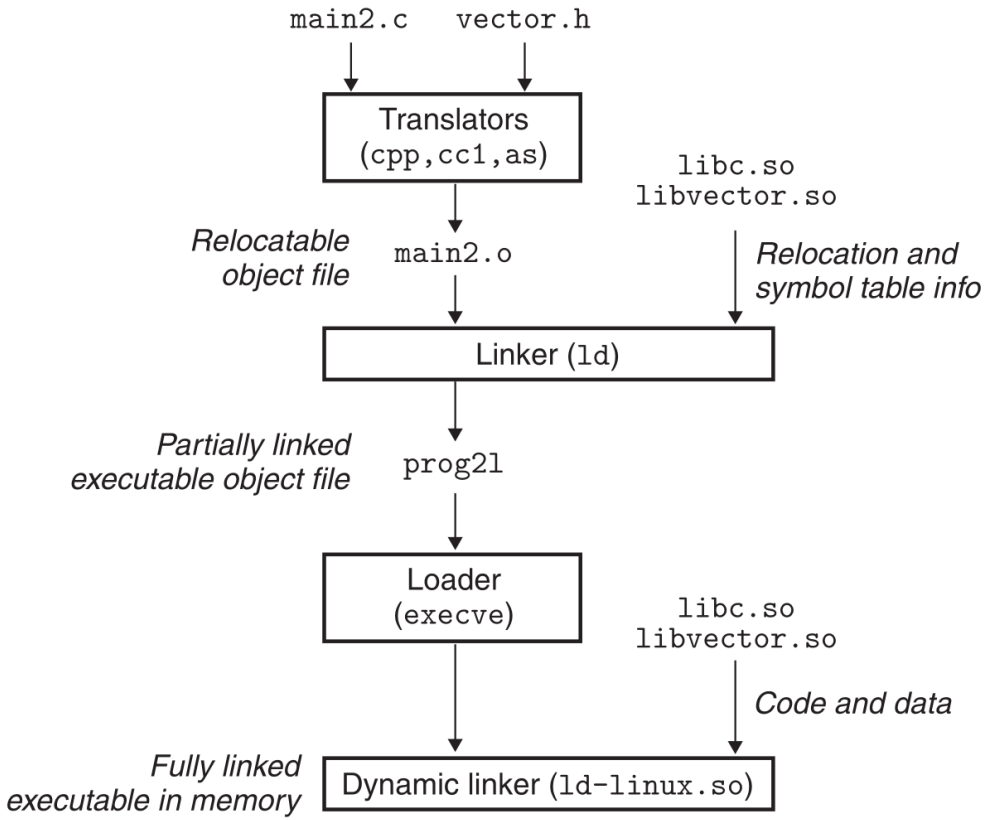
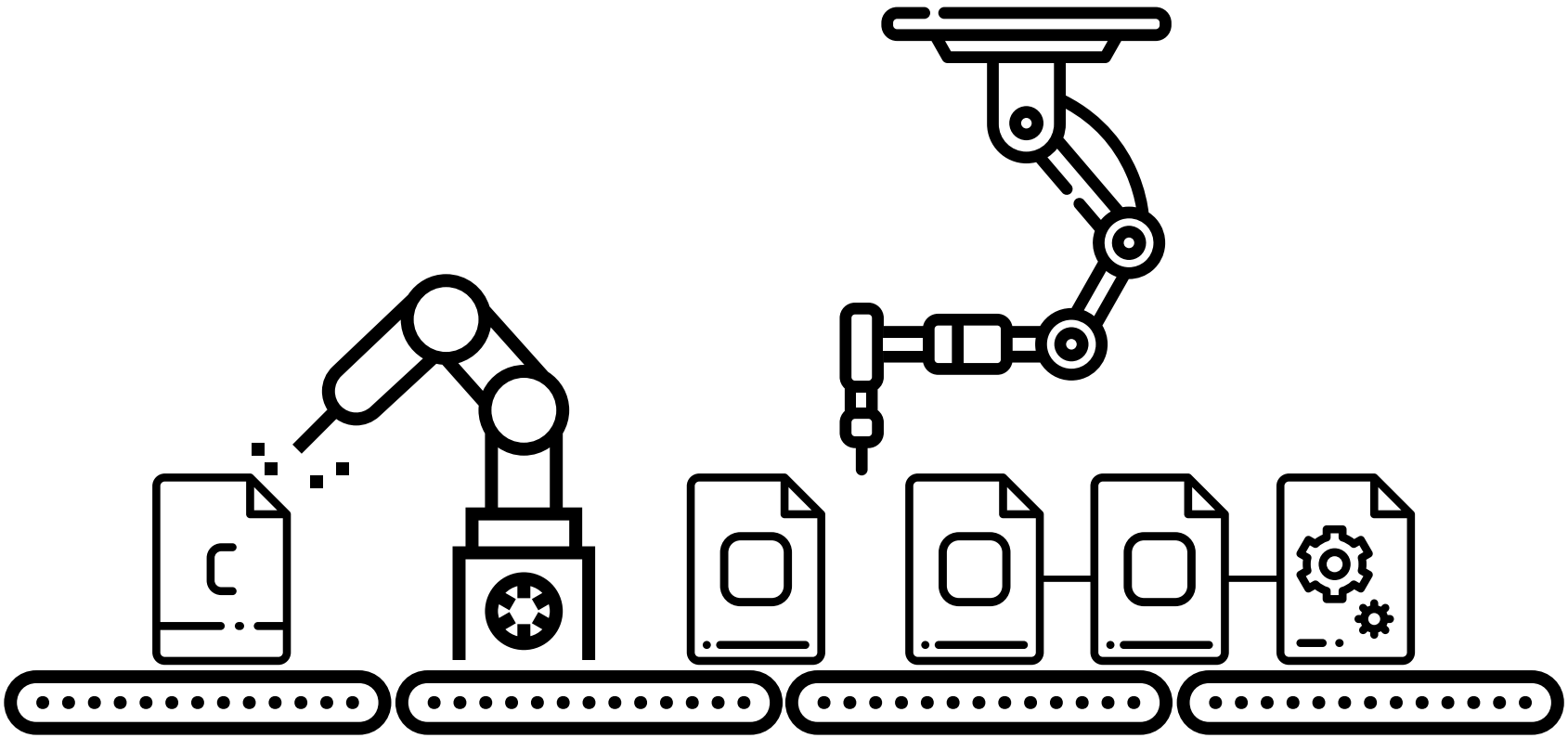


Abbildung: dynamic linking [3]



GCC

Optionen

- gcc main.c sum.c
- gcc main.c sum.c -o main

```c
/* main.c */
int sum(int a, int b);
static int sub(int a, int b) { return a - b; }

int main() {
  int differenz = sub(42, 0);
  int summe = sum(22, 20);
  return 0;
}
```

```c
/* sum.c */
int sum(int a, int b) { return a + b; }
```

```
1  # 0 "main.c"
2  # 0 "<built-in>"
3  # 0 "<command-line>"
4  # 1 "/usr/include/stdc-predef.h" 1 3 4
5  # 0 "<command-line>" 2
6  # 1 "main.c"
7
8  int sum(int a, int b);
9  static int sub(int a, int b) { return a - b; }
10
11 int main() {
12   int differenz = sub(42, 0);
13   int summe = sum(22, 20);
14   return 0;
15 }
```

gcc -E main.c -o main.i

```
24 main:
25 .LFB1:
26   .cfi_startproc
27   pushq %rbp
28   .cfi_def_cfa_offset 16
29   .cfi_offset 6, -16
30   movq  %rsp, %rbp
31   .cfi_def_cfa_register 6
32   subq  $16, %rsp
33   movl  $0, %esi
34   movl  $42, %edi
35   call  sub
36   movl  %eax, -8(%rbp)
37   movl  $20, %esi
38   movl  $22, %edi
39   call  sum@PLT
40   movl  %eax, -4(%rbp)
41   movl  $0, %eax
42   leave
43   .cfi_def_cfa 7, 8
44   ret
```

gcc -S main.c -o main.s

```
00000170: 0000 0000 0000 0000 1200 0000 0000 0000  ... ... ... ... ... .
00000180: 0c00 0000 1200 0100 1200 0000 0000 0000  ... ... ... ... ... .
00000190: 3300 0000 0000 0000 1100 0000 1000 0000  3... ... ... ... ...
000001a0: 0000 0000 0000 0000 0000 0000 0000 0000  ... ... ... ... ... .
000001b0: 2700 0000 1000 0000 0000 0000 0000 0000  '... ... ... ... ...
000001c0: 0000 0000 0000 0000 006d 6169 6e2e 6300  ... ... ...main.c.
000001d0: 7375 6200 6d61 696e 005f 474c 4f42 414c  sub.main._GLOBAL
000001e0: 5f4f 4646 5345 545f 5441 424c 455f 0073  _OFFSET_TABLE_.s
000001f0: 756d 0000 0000 0000 3700 0000 0000 0000  um... ...7... ... .
00000200: 0400 0000 0600 0000 fcff ffff ffff ffff  ... ... ... ... ... .
00000210: 2000 0000 0000 0000 0200 0000 0200 0000   ... ... ... ... ...
00000220: 0000 0000 0000 0000 4000 0000 0000 0000  ... ... ..@... ... .
00000230: 0200 0000 0200 0000 1200 0000 0000 0000  ... ... ... ... ... .
00000240: 002e 7379 6d74 6162 002e 7374 7274 6162  ..symtab..strtab
00000250: 002e 7368 7374 7274 6162 002e 7265 6c61  ..shstrtab..rela
00000260: 2e74 6578 7400 2e64 6174 6100 2e62 7373  .text..data..bss
00000270: 002e 636f 6d6d 656e 7400 2e6e 6f74 652e  ..comment..note.
00000280: 474e 552d 7374 6163 6b00 2e6e 6f74 652e  GNU-stack..note.
00000290: 676e 752e 7072 6f70 6572 7479 002e 7265  gnu.property..re
000002a0: 6c61 2e65 685f 6672 616d 6500 0000 0000  la.eh_frame... ..
000002b0: 0000 0000 0000 0000 0000 0000 0000 0000  ... ... ... ... ... .
```

gcc -c main.c && xxd main.o » hex.txt

- -I *dir*
- -iquote *dir*
- -L *dir*
- -l*name*

- -Wall
- -Werror

Erweiterungen

```c
#include <stdio.h>

void nestedFunction(int a, int b) {
  int add(int a, int b) { return a + b; }

  printf("%d", add(a, b));
}

int main() {
  nestedFunction(23, 46);
  return 0;
}
```

Nested Function

```c
#include <stdio.h>

void caseRanges(int input) {
  switch (input) {
    case 0 ... 10:
      printf("Ich bin %d :)\n", input);
      break;
    case 'A' ... 'Z':
      printf("Ich bin %c :)\n", input);
      break;
  }
}

int main() {
  caseRanges(0b0111);
  caseRanges('E');
  return 0;
}
```

Case Ranges und Binary Constants

Makefile

```
1 main : main.o add.o
2   gcc main.o add.o -o main
3
4 main.o : main.c
5   gcc -c main.c
6
7 add.o : add.c
8   gcc -c add.c
```

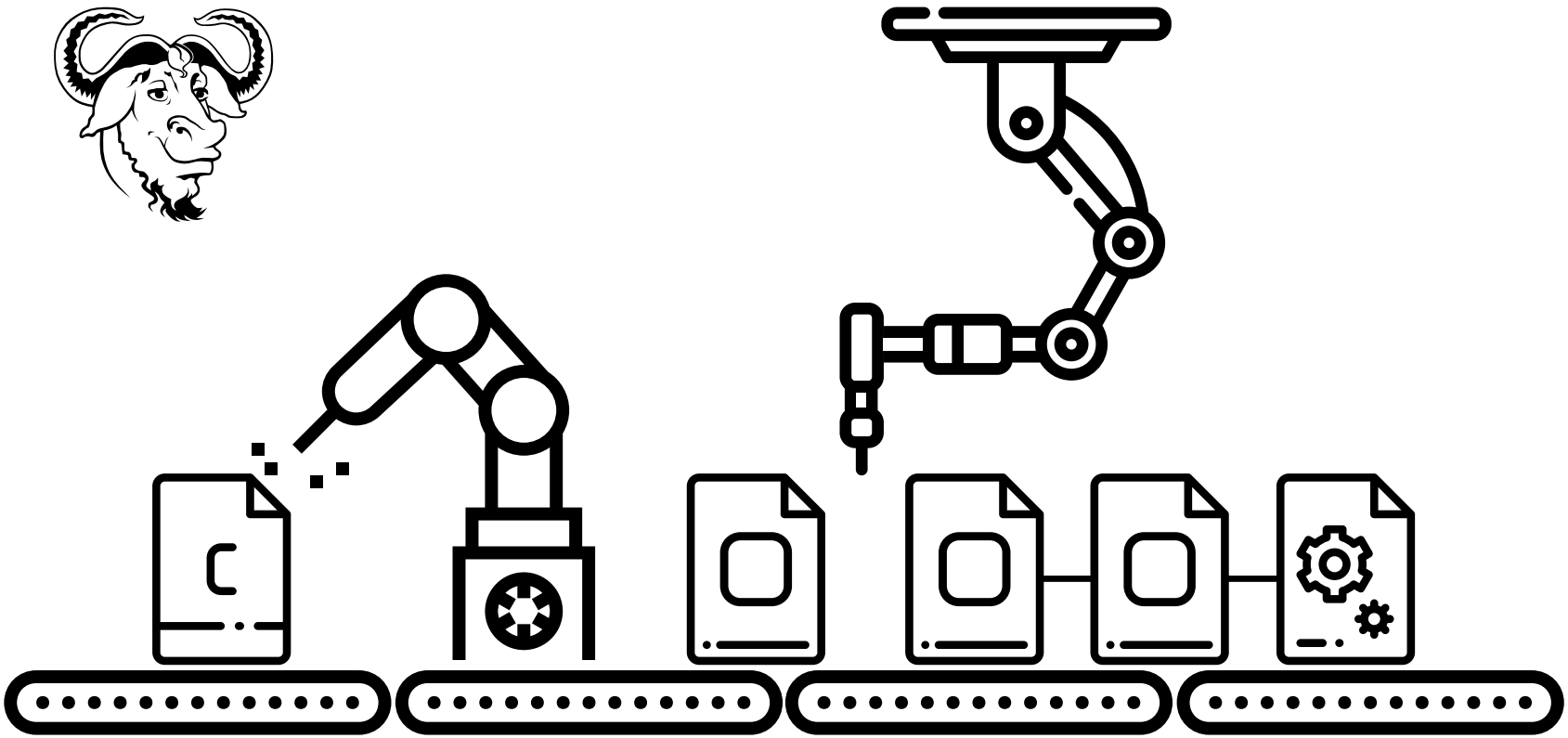
Syntax

```
1  objekte = main.o add.o
2
3  main : $(objekte)
4    gcc $(objekte) -o main
5
6  main.o : main.c
7    gcc -c main.c
8
9  add.o : add.c
10   gcc -c add.c
11
12 clean :
13   rm main $(objekte)
```

Variablen

```
objekte = main.o add.o

main : $(objekte)

main.o : main.c

add.o : add.c

clean :
  rm main $(objekte)
```

Implizite Regeln



Literatur I

[1] Stevanovic, M.: Advanced C and C++ Compiling. Apress (2014). ISBN 978-1-4302-6667-9

[2] Aho, A., Lam, M., Sethi, R., and Ullman, J.: Compilers: Principles, Techniques, and Tools. Addison Wesley, 2 edn. (2006). ISBN 978-0321486813

[3] Bryant, R. E. and O'Hallaron, D. R.: Computer Systems: A Programmer's Perspective. Addison Wesley, 3 edn. (2015). ISBN 978-0134092669

[4] von Hagen, W.: The Definitive Guide to GCC. Apress (2006). ISBN 978-1590595855

Literatur II

[5] Cooper, K. and Torczon, L.: Engineering a Compiler. Morgan Kaufmann, 2 edn. (2011). ISBN 978-0120884780

[6] GNU make. https://www.gnu.org/software/make/manual/make.html#Complex-Makefile. Zuletzt aufgerufen am: 07.06.2021

[7] GNU SVG Vector. https://www.svgrepo.com/svg/306104/gnu. Zuletzt aufgerufen am: 07.06.2021