Systemnahe Programmierung in C

34 Speicherorganisation

J. Kleinöder, D. Lohmann, V. Sieh

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Sommersemester 2025

http://sys.cs.fau.de/lehre/ss25

Speicherorganisation

```
int a;                          // a: global, uninitialized
int b = 1;                      // b: global, initialized
const int c = 2;                // c: global, const

void main(void) {
  static int s = 3;             // s: local, static, initialized
  int x, y;                     // x: local, auto; y: local, auto
  char *p = malloc(100);        // p: local, auto; *p: heap (100 byte)
}
```

Wo kommt der Speicher für diese Variablen her?

Speicherorganisation

```
int a;                          // a: global, uninitialized
int b = 1;                      // b: global, initialized
const int c = 2;                // c: global, const

void main(void) {
  static int s = 3;             // s: local, static, initialized
  int x, y;                     // x: local, auto; y: local, auto
  char *p = malloc(100);        // p: local, auto; *p: heap (100 byte)
}
```

Wo kommt der Speicher für diese Variablen her?

- **Statische Allokation** – Reservierung beim Übersetzen / Linken
  - Betrifft alle globalen/statischen Variablen, sowie den Code ↪ 12–5
  - Allokation durch Platzierung in einer Sektion

.text	– enthält den Programmcode	`main()`
.bss	– enthält alle mit 0 initialisierten Variablen	`a`
.data	– enthält alle mit anderen Werten initalisierten Variablen	`b,s`
.rodata	– enthält alle unveränderlichen Variablen	`c`

Speicherorganisation

```c
int a;                          // a: global, uninitialized
int b = 1;                      // b: global, initialized
const int c = 2;                // c: global, const

void main(void) {
  static int s = 3;             // s: local, static, initialized
  int x, y;                     // x: local, auto; y: local, auto
  char *p = malloc(100);        // p: local, auto; *p: heap (100 byte)
}
```

Wo kommt der Speicher für diese Variablen her?

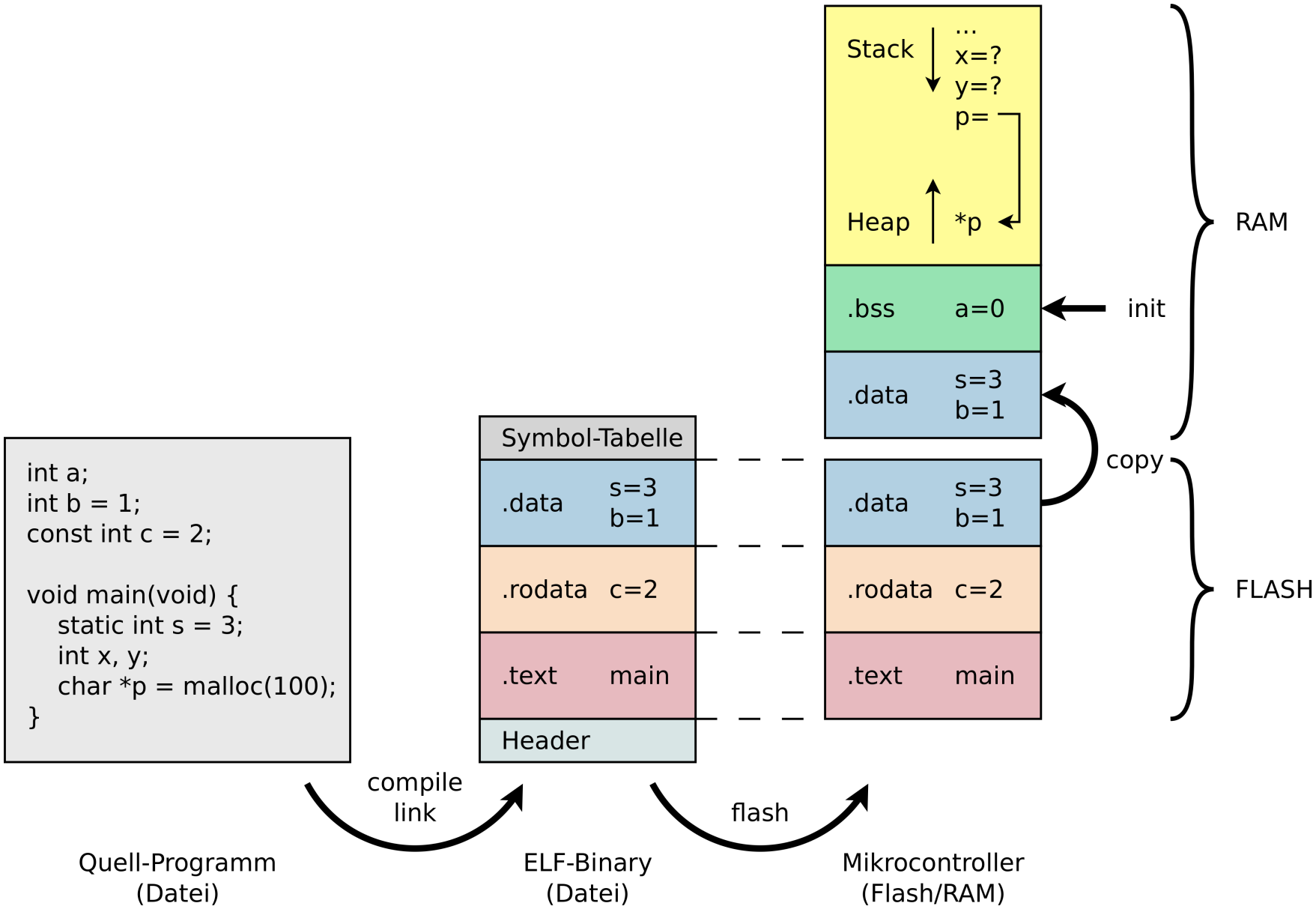
- **Statische Allokation** – Reservierung beim Übersetzen / Linken
  - Betrifft alle globalen/statischen Variablen, sowie den Code ↪ 12–5
  - Allokation durch Platzierung in einer Sektion

.text	– enthält den Programmcode	`main()`
.bss	– enthält alle mit 0 initialisierten Variablen	`a`
.data	– enthält alle mit anderen Werten initalisierten Variablen	`b,s`
.rodata	– enthält alle unveränderlichen Variablen	`c`

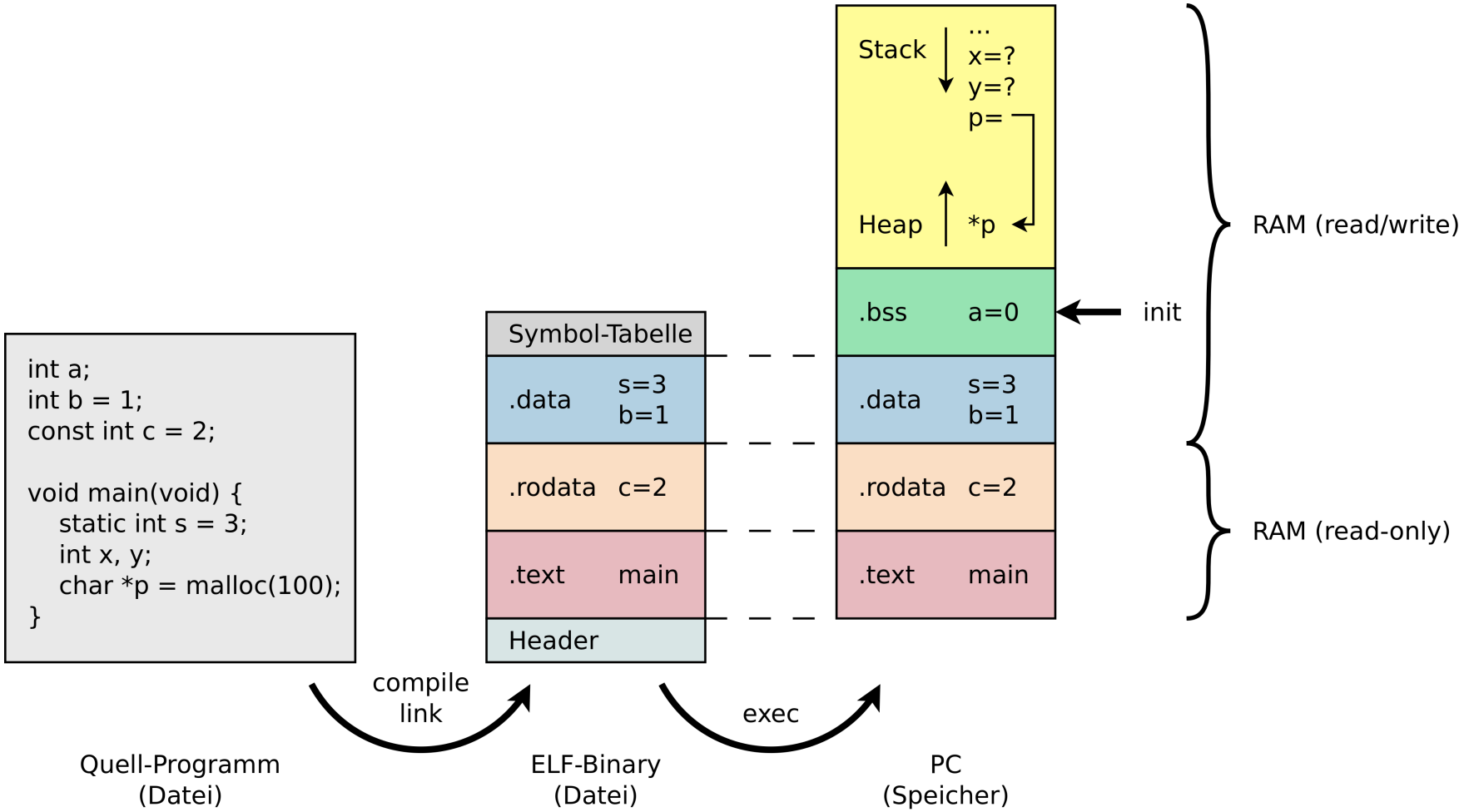
- **Dynamische Allokation** – Reservierung zur Laufzeit
  - Betrifft lokale auto-Variablen und explizit angeforderten Speicher

Stack	– enthält alle aktuell lebendigen auto-Variablen	`x,y,p`
Heap	– enthält explizit mit `malloc()` angeforderte Speicherbereiche	`*p`

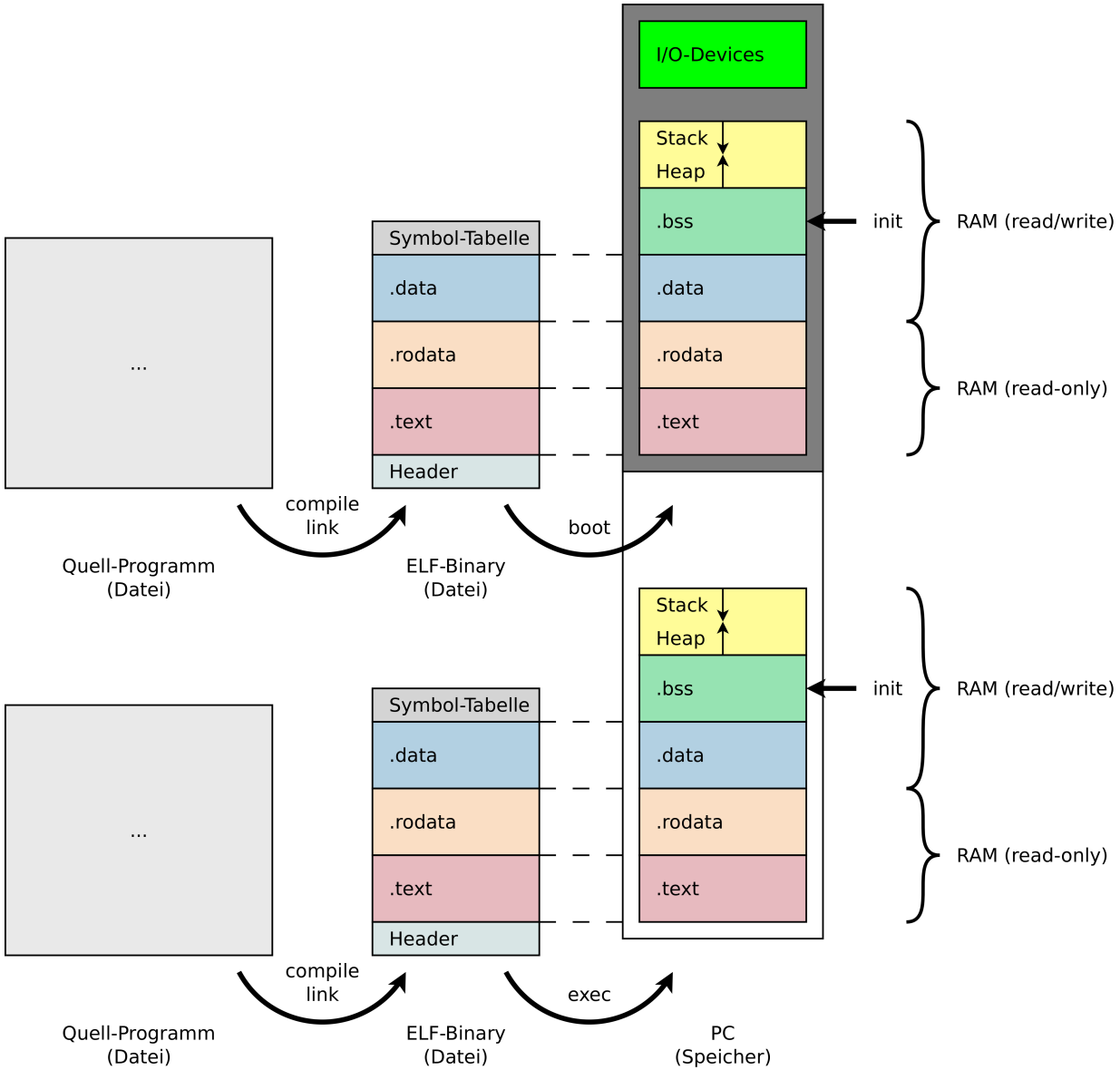
Speicherorganisation auf einem μC



Speicherorganisation mit Betriebssystem



Speicherorganisation mit Betriebssystem (Forts.)



Speicherorganisation mit Betriebssystem (Forts.)