

Systemnahe Programmierung in C (SPiC)

34 Speicherorganisation

Jürgen Kleinöder, Daniel Lohmann, Volkmar Sieh

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Sommersemester 2020

http://www4.cs.fau.de/Lehre/SS20/V_SPiC



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?



```
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
- Allokation durch Platzierung in einer **Sektion**

↪ 12-5

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



```
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
- Allokation durch Platzierung in einer **Sektion**

↪ 12-5

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

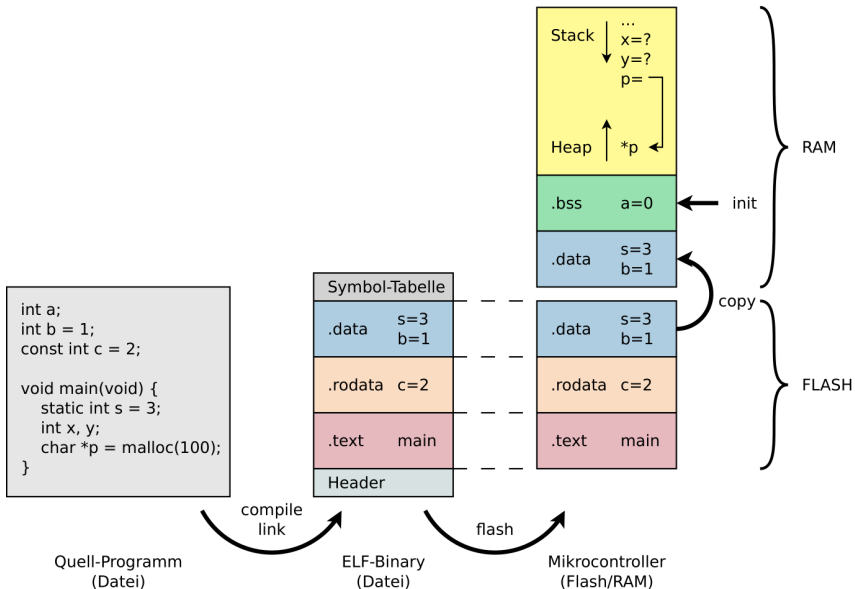
■ Dynamische Allokation – Reservierung zur Laufzeit

- Betrifft lokale auto-Variablen und explizit angeforderten Speicher

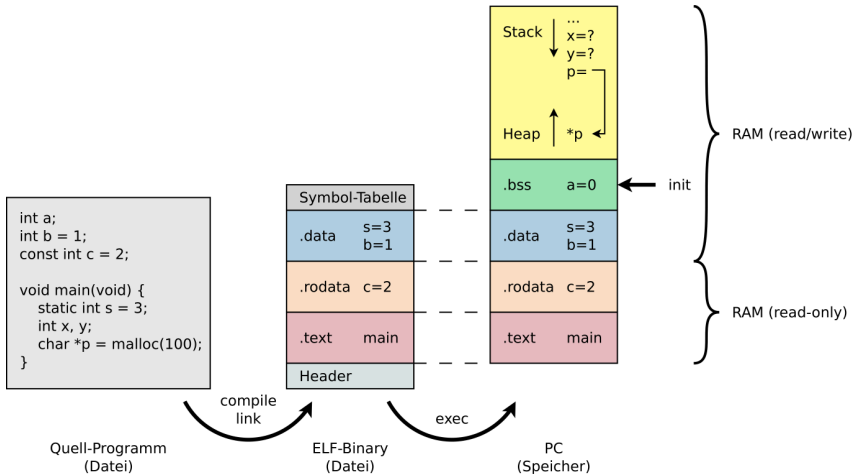
Stack	– enthält alle aktuell lebendigen auto-Variablen	<code>x,y,p</code>
Heap	– enthält explizit mit <code>malloc()</code> angeforderte Speicherbereiche	<code>*p</code>



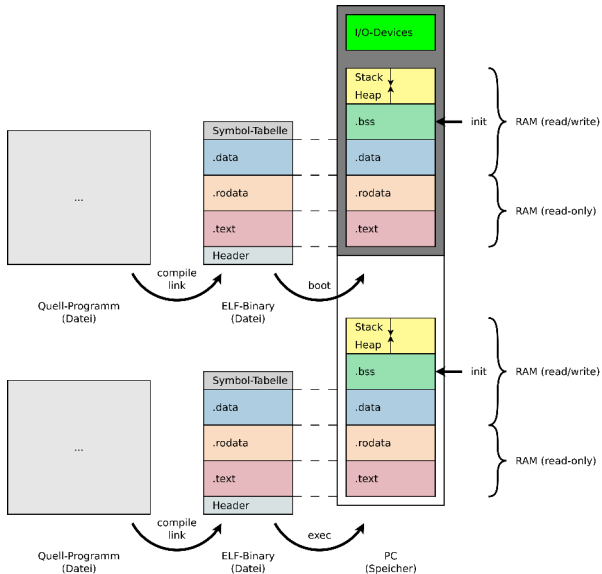
Speicherorganisation auf einem μC



Speicherorganisation mit Betriebssystem



Speicherorganisation mit Betriebssystem (Forts.)



Speicherorganisation mit Betriebssystem (Forts.)

