System-Level Programming

36 Organization of Memory – Summary

Peter Wägemann

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU)

Summer Term 2025

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

Static vs. Dynamic Allocation

- For **µC development** static allocation is preferred
  - **advantage:** The required memory is already known during compilation / linking (can be determined with `size`/`avr-size` command)
  - memory-limit violations detected upfront (memory is scarce! ↪ 1–4)

```
~> size sections.avr
text     data     bss     dec     hex filename
682      10       6       698     2ba sections.avr
```

Sizes of the sections of the program ↪ 34–1

⇝ When possible, memory should be allocated with `static` variables
  - always consider the rule of narrowest scope ↪ 12–6
  - always apply the rule of shortest possible "reasonable" lifespan

- In comparison, a heap is more expensive ⇝ should be avoided
  - additional costs in memory for management structures and code
  - memory required during runtime complicated to estimate
  - risk of memory leaks and programming errors

Static vs. Dynamic Allocation (continued)

- When developing for an **operating-system platform** it can be sensible to use dynamic allocation
  - **advantage:** dynamic adaption to the size of the input data (e. g., for strings)
  - reduced risk of *buffer-overflow attacks*

⇝ If possible, allocate memory for input data on the heap

- still, the risk of programming errors and memory leaks remains