System-Level Programming

36 Organisation of Memory – Summary

J. Kleinöder, D. Lohmann, V. Sieh, P. Wägemann

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Erlangen-Nürnberg

Summer Term 2024

http://sys.cs.fau.de/lehre/ss24

36-Speicher-Zusammenfassung_en

Static vs. Dynamic Allocation

- For μ**C development** static allocation is preferred
  - **Advantage:** The required memory is already known during compilation / linking (can be returned with `size`/`avr-size` command)
  - Problems with memory limits are 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!