System-Level Programming

1 Introduction

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
Introduction

Deepen

- knowledge of concepts and techniques of computer science and software development
- Starting point: Algorithms, Programming, and Data Representation
- Main focus: System-Level Programming (SLP) in C

Development

- of software in C for a μController (μC) and an operating-system platform (Linux)
- SPiCboard learning development platform with an ATmega-μC
- Practical experience in hardware and system-level software development

Understanding

- of technological language and hardware basics for the development of system-level software
  - Being able to understand and assess the language C and
  - Dealing with concurrency and hardware orientation
  - Dealing with the abstractions of an operating system (files, processes, . . .)
Motivation: Embedded Systems

Omnipresent: 98–99 percent of processors are being used in embedded systems [7].

Cost-sensitive: 70–80 percent of all produced processors are DSPs and µController, based on 8-bit architecture or lower [7, 8].

Relevant: 25 percent of job offers for EE engineers do contain the terms embedded or automotive (http://stepstone.com).
Motivation: Embedded Systems

Omnipresent: 98–99 percent of processors are being used in embedded systems [7]
Motivation: Embedded Systems

- Omnipresent: 98–99 percent of processors are being used in embedded systems [7]
- Cost-sensitive: 70–80 percent of all produced processors are DSPs and µController, based on 8-bit architecture or lower [7, 8]
Motivation: Embedded Systems

- **Omnipresent:** 98–99 percent of processors are being used in embedded systems [7]

- **Cost-sensitive:** 70–80 percent of all produced processors are DSPs and μController, based on 8-bit architecture or lower [7, 8]

- **Relevant:** 25 percent of job offers for EE engineers do contain the terms embedded or automotive (http://stepstone.com)
Motivation: Embedded Systems

Source: IC Insights 2014 McClean Report
Motivation: The ATmega-µC Family (8-bit)

<table>
<thead>
<tr>
<th>Type</th>
<th>Flash</th>
<th>SRAM</th>
<th>IO</th>
<th>Timer 8/16</th>
<th>UART</th>
<th>SPI</th>
<th>ADC</th>
<th>PWM</th>
<th>EUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTINY13</td>
<td>1 KiB</td>
<td>64 B</td>
<td>6</td>
<td>1/-</td>
<td>-</td>
<td>-</td>
<td>1*4</td>
<td>-</td>
<td>2,20</td>
</tr>
<tr>
<td>ATTINY2313</td>
<td>2 KiB</td>
<td>128 B</td>
<td>18</td>
<td>1/1</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2,99</td>
</tr>
<tr>
<td>ATMEGA48</td>
<td>4 KiB</td>
<td>512 B</td>
<td>23</td>
<td>2/1</td>
<td>1</td>
<td>1</td>
<td>8*10</td>
<td>6</td>
<td>2,40</td>
</tr>
<tr>
<td>ATMEGA16</td>
<td>16 KiB</td>
<td>1024 B</td>
<td>32</td>
<td>2/1</td>
<td>1</td>
<td>1</td>
<td>8*10</td>
<td>4</td>
<td>6,40</td>
</tr>
<tr>
<td>ATMEGA32</td>
<td>32 KiB</td>
<td>2048 B</td>
<td>32</td>
<td>2/1</td>
<td>1</td>
<td>1</td>
<td>8*10</td>
<td>4</td>
<td>5,40</td>
</tr>
<tr>
<td>ATMEGA64</td>
<td>64 KiB</td>
<td>4096 B</td>
<td>53</td>
<td>2/2</td>
<td>2</td>
<td>1</td>
<td>8*10</td>
<td>8</td>
<td>–</td>
</tr>
<tr>
<td>ATMEGA128</td>
<td>128 KiB</td>
<td>4096 B</td>
<td>53</td>
<td>2/2</td>
<td>2</td>
<td>1</td>
<td>8*10</td>
<td>8</td>
<td>19,80</td>
</tr>
<tr>
<td>ATMEGA256</td>
<td>256 KiB</td>
<td>8192 B</td>
<td>86</td>
<td>2/2</td>
<td>4</td>
<td>1</td>
<td>16*10</td>
<td>16</td>
<td>15,50</td>
</tr>
</tbody>
</table>

ATmega variants (selection) and market prices (Reichelt Elektronik, April 2023)
Motivation: The ATmega-μC Family (8-bit)

<table>
<thead>
<tr>
<th>Type</th>
<th>Flash</th>
<th>SRAM</th>
<th>IO</th>
<th>Timer 8/16</th>
<th>UART</th>
<th>SPI</th>
<th>ADC</th>
<th>PWM</th>
<th>EUR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTINY13</td>
<td>1 KiB</td>
<td>64 B</td>
<td>6</td>
<td>1/-</td>
<td>-</td>
<td>-</td>
<td>1*4</td>
<td>-</td>
<td>2,20</td>
</tr>
<tr>
<td>ATTINY2313</td>
<td>2 KiB</td>
<td>128 B</td>
<td>18</td>
<td>1/1</td>
<td>-</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>2,99</td>
</tr>
<tr>
<td>ATMEGA48</td>
<td>4 KiB</td>
<td>512 B</td>
<td>23</td>
<td>2/1</td>
<td>1</td>
<td>1</td>
<td>8*10</td>
<td>6</td>
<td>2,40</td>
</tr>
<tr>
<td>ATMEGA16</td>
<td>16 KiB</td>
<td>1024 B</td>
<td>32</td>
<td>2/1</td>
<td>1</td>
<td>1</td>
<td>8*10</td>
<td>4</td>
<td>6,40</td>
</tr>
<tr>
<td>ATMEGA32</td>
<td>32 KiB</td>
<td>2048 B</td>
<td>32</td>
<td>2/1</td>
<td>1</td>
<td>1</td>
<td>8*10</td>
<td>4</td>
<td>5,40</td>
</tr>
<tr>
<td>ATMEGA64</td>
<td>64 KiB</td>
<td>4096 B</td>
<td>53</td>
<td>2/2</td>
<td>2</td>
<td>1</td>
<td>8*10</td>
<td>8</td>
<td>–</td>
</tr>
<tr>
<td>ATMEGA128</td>
<td>128 KiB</td>
<td>4096 B</td>
<td>53</td>
<td>2/2</td>
<td>2</td>
<td>1</td>
<td>8*10</td>
<td>8</td>
<td>19,80</td>
</tr>
<tr>
<td>ATMEGA256</td>
<td>256 KiB</td>
<td>8192 B</td>
<td>86</td>
<td>2/2</td>
<td>4</td>
<td>1</td>
<td>16*10</td>
<td>16</td>
<td>15,50</td>
</tr>
</tbody>
</table>

ATmega variants (selection) and market prices (Reichelt Elektronik, April 2023)

Becomes visible: **resource scarcity**

- **Flash** (storage for program code and constant data) is **scarce**
- **RAM** (storage for runtime variables) is **extremely scarce**
- few bytes “wasted” \(\sim\) significantly higher cost per piece
Motivation: C as a Language

- System-level software development predominantly takes place in C.
- Why C? (and not Python/Java/Scala/<favourite language>)

- Runtime efficiency (CPU)
  - Translated C code runs on the processor directly
  - No checks for programming errors at runtime

- Space efficiency (storage)
  - Code and data can be stored rather compact
  - No checks for data access at runtime

- Immediacy (machine orientation)
  - C allows for direct access to storage and registers

- Portability
  - There is a C compiler for every platform

C was "invented" (1973), to implement the OS UNIX portable [4, 6]; C is the lingua franca of system-level programming!
Motivation: C as a Language

- System-level software development predominantly takes place in C.
  - **Why C?** (and not Python/Java/Scala/<favourite language>)

- C stands for a multitude of important features
  - Runtime efficiency (CPU)
    - Translated C code runs on the processor directly
    - No checks for programming errors at runtime
Motivation: C as a Language

- System-level software development predominantly takes place in C.
  - **Why C?** (and not Python/Java/Scala/<favourite language>)

- C stands for a multitude of important features
  - Runtime efficiency (CPU)
    - Translated C code runs on the processor directly
    - No checks for programming errors at runtime
  - Space efficiency (storage)
    - Code and data can be stored rather compact
    - No checks for data access at runtime
Motivation: C as a Language

- System-level software development predominantly takes place in C.
  - **Why C?** (and not Python/Java/Scala/<favourite language>)

C stands for a multitude of important features

- Runtime efficiency (CPU)
  - Translated C code runs on the processor directly
  - No checks for programming errors at runtime

- Space efficiency (storage)
  - Code and data can be stored rather compact
  - No checks for data access at runtime

- Immediacy (machine orientation)
  - C allows for direct access to storage and registers
Motivation: C as a Language

- System-level software development predominantly takes place in C.
  - **Why C?** (and not Python/Java/Scala/<favourite language>)

- C stands for a multitude of important features
  - Runtime efficiency (CPU)
    - Translated C code runs on the processor directly
    - No checks for programming errors at runtime
  - Space efficiency (storage)
    - Code and data can be stored rather compact
    - No checks for data access at runtime
  - Immediacy (machine orientation)
    - C allows for direct access to storage and registers
  - Portability
    - There is a C compiler for **every** platform
    - C was “invented” (1973), to implement the OS UNIX portable [4, 6]
System-level software development predominantly takes place in **C**.

**Why C?** (and not Python/Java/Scala/<favourite language>)

C stands for a multitude of important features

- **Runtime efficiency (CPU)**
  - Translated C code runs on the processor directly
  - No checks for programming errors at runtime

- **Space efficiency (storage)**
  - Code and data can be stored rather compact
  - No checks for data access at runtime

- **Immediacy (machine orientation)**
  - C allows for direct access to storage and registers

- **Portability**
  - There is a C compiler for **every** platform
  - C was “invented” (1973), to implement the OS UNIX portable [4, 6]

\[C\] is the **lingua franca** of system-level programming!
Motivation: SLP – Syllabus and Concept

- **Teaching objective:** system-level programming in C
  - This is a really broad field: hardware programming, operating systems, middleware, data bases, distributed systems, compiler construction, ... 
  - Additionally, we have the goal of learning the language C itself

- **Approach**
  - Concentration on two domains
    - \( \mu \)C programming
    - Software development for Linux system interface
  - Experience contrast \( \mu \)C-environment \( \leftrightarrow \) operating system
  - Concepts and techniques get teachable and tangible with the help of various examples
  - **High relevance** for the target audience (EE, ME, ...)

© klsw System-Level Programming (ST 24) 1 Introduction – Why C?
Motivation: SLP

At the end of the lecture, everyone should be able to assess,
what a μController can (not) do,
how labor-intensive & beneficial its programming is,
what an operating system does (not) provide,
how labor-intensive & beneficial it is, to use one.

Everyone should be able to work with a computer scientist, if necessary...
This handout of the lecture notes will be provided online.
- Chapters are available as individual files
- The handout contains (some) additional information

However, the handout cannot be used as a substitute for making your own notes!
Literature Recommendations

[3] Recommended for Beginners:


[5] The “classic” (more suitable as a reference):

References


