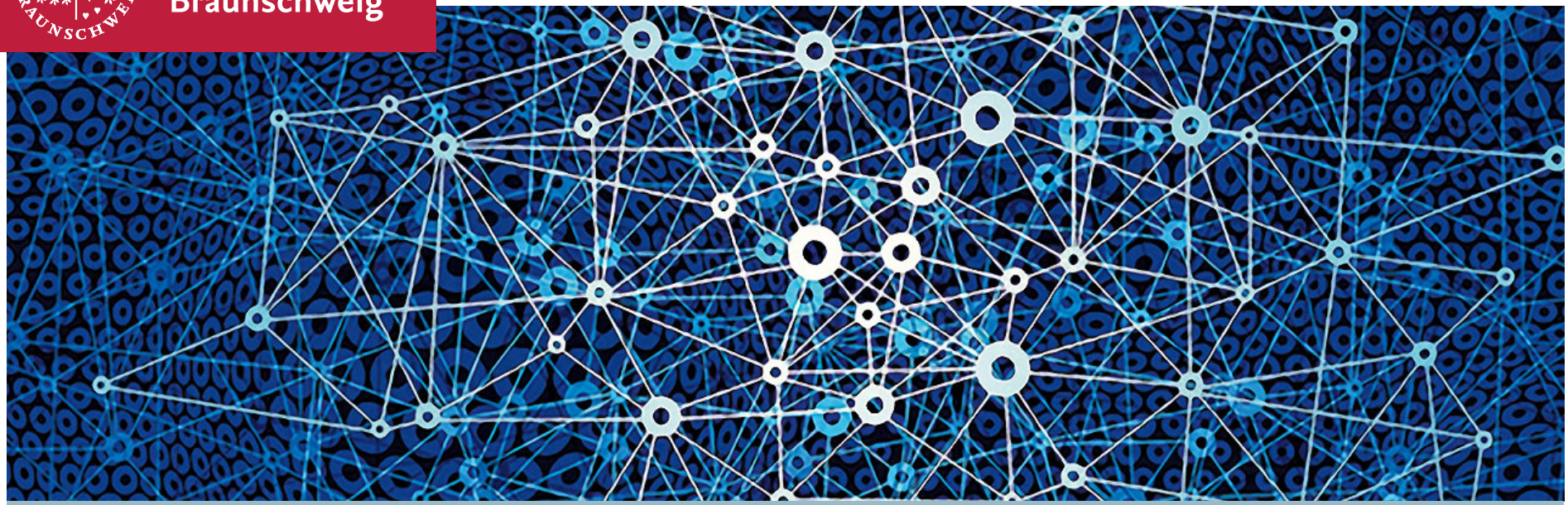


Systems programming

Seminar Distributed Systems

Ines Messadi, April 9, 2019

Technische Universität Braunschweig, IBR

Today's Overview

- **Seminar Overview & Organization**
- **Seminar Topics & Assignment**

Agenda

- **Seminar Overview & Organization**
- **Seminar Topics & Assignment**

Purpose of this seminar

- **Topics**
  - Systems programming in a well-established environement
  - Modern hardware & tools
  - Problems of programming

  → Practical Seminar

- **Goals**
  - Learn something new about systems programming
  - Learn to write about technical work
  - Learn to give a good talk
  - Get some practice in LaTeX

  → Preparation for bachelor and master thesis

Organisation

- Essay and presentation in **English**
  - Slides *and* voice track

- **Regular : Every Tuesday, 13:15, Room 105**
  - Attendance is mandatory, except under exceptional circumstances
  - If you can't attend, write an email

- **Contact & all submissions** → `messadi@ibr.cs.tu-bs.de`

Workload

- Presentation to tell us what you learn $\approx$ 25-30 min
  - Present results, or a demo
  - Answer some general questions
- Programming assignment & Reading materials
- Brief essay (4-5 pages)
- Review and feedback
  - Attend dry-run
  - Constructive written feedback
- Active participation
  - Asking questions & contributing to the discussion
- Attend all talks

Presentation

- **Presentation elements**
  - Why is this interesting?
  - Examples of applications
  - What is the current status of research?
  - What are the key components of my approach and results?

- **Templates & Submission**
  - Templates: `https://www.ibr.cs.tu-bs.de/kb/templates.html`
  - Put title, author, and page numbers on each slide

Presentation best practices

- Rules of thumb
  - Your first slide should **not** be the outline!
  - Your last slide should be the conclusion

- Recommendations
  - One meaningful picture on each slide
    $\rightarrow$ Draw your own figures if possible
  - **Emphasize** important things
  - Omit line breaks
  - Start with uppercase

Debugging your essay

- Rules of Thumb
  - Simple sentences are effective
  - In your own words
  - Use a spell checker → `www.grammarly.com`
  - Have other people read it – how? → Essay feedback
  - Feedback is a suggestion for improvement

- Citations
  - A citation is an annotation for a sentence
  - If you remove the citation, the sentence should still be grammatically correct and complete
  - Example: "[A072] contains a definition of..." → Wrong

Essay recommended structure

1. Abstract
2. Introduction & Motivation
3. Content
4. Related Work
5. Conclusion
6. References

Not needed:

- Appendix

SIG Proceedings Paper in LaTeX Format*

Extended Abstract†

Ben Trovato‡
Institute for Clarity in Documentation
Dublin, Ohio
trovato@corporation.com

G.K.M. Tobin§
Institute for Clarity in Documentation
Dublin, Ohio
webmaster@marysville-ohio.com

Lars Thørväld¶
The Thørväld Group
Hekla, Iceland
larst@affiliation.org

Valerie Béranger
Inria Paris-Rocquencourt
Rocquencourt, France

Aparna Patel
Rajiv Gandhi University
Doimukh, Arunachal Pradesh, India

Huifen Chan
Tsinghua University
Haidian Qu, Beijing Shi, China

Charles Palmer
Palmer Research Laboratories
San Antonio, Texas
cpalmer@prl.com

John Smith
The Thørväld Group
jsmith@affiliation.org

Julius P. Kumquat
The Kumquat Consortium
jpkumquat@consortium.net

ABSTRACT

This paper provides a sample of a LaTeX document which conforms, somewhat loosely, to the formatting guidelines for ACM SIG Proceedings.[1]

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability;

KEYWORDS

ACM proceedings, LaTeX, text tagging

ACM Reference Format:
Ben Trovato, G.K.M. Tobin, Lars Thørväld, Valerie Béranger, Aparna Patel, Huifen Chan, Charles Palmer, John Smith, and Julius P. Kumquat. 1997. SIG Proceedings Paper in LaTeX Format: Extended Abstract. In *Proceedings of ACM Woodstock conference (WOODSTOCK'97)*, Jennifer B. Sartor, Theo D'Hondt, and Wolfgang De Meuter (Eds.). ACM, New York, NY, USA, Article 4, 5 pages. https://doi.org/10.475/123_4

1 INTRODUCTION

The *proceedings* are the records of a conference.[2] ACM seeks to give these conference by-products a uniform, high-quality appearance. To do this, ACM has some rigid requirements for the format of the proceedings documents: there is a specified format (balanced double columns), a specified set of fonts (Arial or Helvetica and Times Roman) in certain specified sizes, a specified live area, centered on the page, specified size of margins, specified column width and gutter size.

2 THE BODY OF THE PAPER

Typically, the body of a paper is organized into a hierarchical structure, with numbered or unnumbered headings for sections, subsections, sub-subsections, and even smaller sections. The command `\section` that precedes this paragraph is part of such a hierarchy.[3] LaTeX handles the numbering and placement of these headings for you, when you use the appropriate heading commands around the titles of the headings. If you want a sub-subsection or smaller part to be unnumbered in your output, simply append an asterisk to the command name. Examples of both numbered and unnumbered headings will appear throughout the balance of this sample document.

Because the entire article is contained in the **document** environment, you can indicate the start of a new paragraph with a blank line in your input file; that is why this sentence forms a separate paragraph.

2.1 Type Changes and *Special* Characters

We have already seen several typeface changes in this sample. You can indicate italicized words or phrases in your text with the command `\textit`; emboldening with the command `\textbf` and typewriter-style (for instance, for computer code) with `\texttt`. But remember, you do not have to indicate typestyle changes when such changes are part of the *structural* elements of your article; for instance, the heading of this subsection will be in a sans serif[4] typeface, but that is handled by the document class file. Take care

*Produces the permission block, and copyright information
†The full version of the author's guide is available as `acmart.pdf` document
‡Dr. Trovato insisted his name be first.
§The secretary disavows any knowledge of this author's actions.
¶This author is the one who did all the really hard work.
[1]This is an abstract footnote
[2]This is a footnote
[3]This is a footnote.
[4]Another footnote here. Let's make this a rather long one to see how it looks.

ACM ISBN 123-4567-24-567/08/06.
https://doi.org/10.475/123_4

Programming assignment

- Specific task for each topic → Build a small prototype
- Own git repository
- Present your own implementation
- Feedback → Improved code quality

- Recommendations
  - Working implementation & tests to run
  - Documented (README & code)
  - Can be run by others
  - No code duplication

Feedback

- Each will have to assist a second student

- This means
  - You will have to **attend dry-run** and give feedback
  - Review the essay and provide feedback within one week

- Reviews are based on a *review template*
  $\rightarrow$ Missing assistance will impact your grade

Procedure & timeline

- Week t-4: Read about topic (Reading materials on web page)
- Week t-3: Start implementing
- Week t-2: Meeting to Discuss topic and approach
- Week t-1: Dry-run $\rightarrow$ presentation must be in a finished state
- Week t: **Presentation of your topic**
- Week t+1: Submission of essay for review
- Week t+2: You receive feedback from a student and supervisor
- Week t+3: Final submission of essay

$\rightarrow$You are responsible for your deadlines

How to have the best possible grade?

1. Programming assignment and presentation **completed and presented successfully**
2. Take into consideration given feedback
3. Be ready for the dry-run with the complete presentation
4. Respect deadlines
5. Attend all talks & active participation
6. Respect the recommendations in slide 6-11

$\Rightarrow$ Don't miss any of these

Agenda

- Seminar Overview & Organization
- **Seminar Topics & Assignment**

Seminar Topic

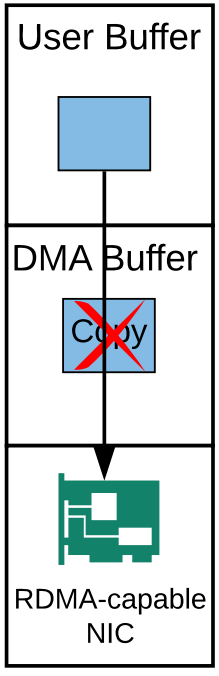
- Topics focus on:
  - Modern Hardware: RDMA, Transactional memory
  - Trusted Execution: e.g SGX
  - Programming languages: e.g Rust
  - OS examples: Linux drivers
- Reading materials on the web page
- In total **8 topics**

Topic 1: Using RDMA efficiently

- RDMA: Remote Direct Memory Access
- CPU **bypassing** technology
  - Ultra **low** latency
  - Ultra **high** throughput
- Orders of magnitude improvements on distributed applications

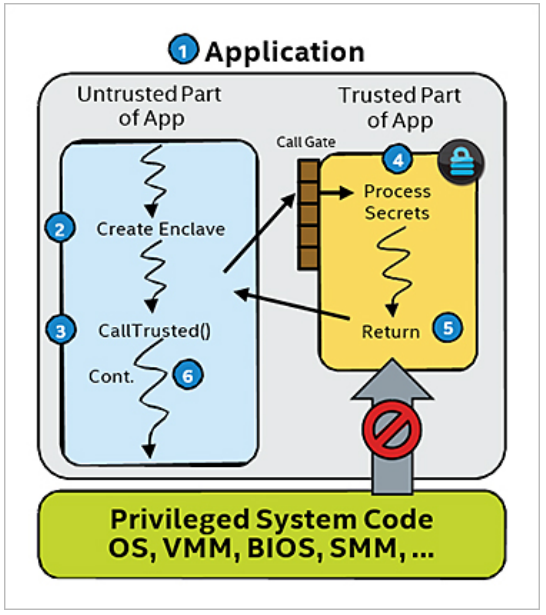
Task:

- Implement a simple key-value store using one-sided operations
- Using the simulation library



Topic 2: Securing applications using SGX

- Intel SGX enclaves are secure isolated compartments
- Protection of execution code within the enclave



Task:

- Use Intel SGX to create a password manager to protect authentication credentials in the pluggable authentication module (PAM)

Topic 3: Using CRDTs

- Conflict-free Replicated Data Types
- A data structure that can be updated without expensive synchronization
- Merge concurrent modification
- Already used in some examples like Redis, Riak, Facebook

Task:

- Explore different CRDTs and show their benefits compared to traditional solutions
- Present a demo, showing CRDTs properties

Topic 4: Lessons Learned from Rust

- Practical and safe alternative to C
- As fast as C/C++
- Concurrent programming language

Task:

- Explain what features Rust provides
- Implement a multi threaded command line tool, e.g grep and if it can usable with other cmd tools

Topic 5: Understanding Transactional Memory

- Transactions instead of locks
- No parallelism issues e.g,deadlocks, race conditions
- Perfect scaling

Task:

- Implement a merkle hash tree using TM, or find an implementation and re-write it.
- Present a hands-on result from using a TM library, and explain its benefits over the usual mutex and locks mechanisms.

Topic 6: How to implement a Linux device driver?

- Mechanisms to users to access particular protected part of hardware
- Enable operating systems and other computer programs to access hardware
- How we can implement a linux device driver?

Tasks:

- Explain how it works
- Implement calculator using ioctl of linux device driver
- Present hands-on results of how we can implement a linux device driver

Topic 7: Atomic operations, from basics to advanced

- Mutual exclusion prevents data races
- Alternatives to locks, but do not suffer from deadlocks
- Relatively quick

Task:

- Present hands-on results explaining atomic operations from basics to advanced
- Explain locks and atomic operations
- Implement examples that gives a performance comparison

Topic 8: Fundamental operating system services

- Present the fundamentals to implement operating system services

Papers:

- Present educational OS
- Choose an example and explain its support for scheduling
- Present a demo

Topic Assignment

Random Topic Assignment by Preference