Project Topics
Common requirements

for suggested and individual project ideas:

- implementation of a **distributed algorithm**
- testing/presentation can be done locally, on one or two machines
- the implementation needs to support multiple machines connected over a network regardless of the environment used for testing/presentation
1. Actor model

- Specify and implement a Rust library implementing the concepts of the actor model.
- Demonstrate the library with a proof-of-concept application of your choice.

References

- Actor model, Wikipedia
- Actors, Enc. Par. Comp. (access through university library)
2. P2P network

- Specify and implement a Rust library which enables the creation of a peer-to-peer network.
- Explain your choices regarding routing and resource discovery.
- Demonstrate the peer-to-peer network with a proof-of-concept application of your choice.

References
- Peer-to-peer, Wikipedia
- Peer-to-Peer, Enc. Par. Comp. (access through university library)
3. Distributed key-value database

- Specify and implement a Rust library which provides a distributed key-value database.
- The data should reside in main memory (RAM).
- Explain your choice regarding the underlying communication model (client/server, actor, P2P, ...).
- Demonstrate your implementation with a proof-of-concept application of your choice.

References

- Key-value database, Wikipedia
- Distributed data store, Wikipedia
4. Software Distributed Shared Memory (SDSM)

- Specify and implement a Rust library which provides a **global view to local memory** across multiple machines.
- Explain the **integration** of the library into the system (for example, a library with user-visible API, integration of the library as a Rust memory allocator, library that can be preloaded, a kernel patch enhancing the Linux memory subsystem, etc.)
- Test and demonstrate your implementation with a **proof-of-concept application** of your choice, for example by instantiating a data structure larger than locally available memory.

**References**

- Distributed shared memory, Wikipedia
- Software Distributed Shared Memory, Enc. Par. Comp (access through university library)
- B-tree, Wikipedia
Your own project

- conforming to the **common requirements** stated earlier
- with a similar amount of work as the suggested topics
- briefly **announce** the general idea of the project until Dec 19 — no written text required
Some application ideas
Suggestions how to use crates.io as a data source:

- perform **queries on crate meta-data**, such as a string search in crate name, crate author, library name
- **full-text search** on crate README files or source comments
- search for **function names** or trait names returning file name and line number for a number of crates — a similar functionality is provided by Haskell’s Hoogle server
- provide a simple **Continuous Integration (CI)** server compiling crates as they are uploaded
Content delivery

- message transport such as **instant messaging**, or a simple **mail** system
- share **files** by propagating individual chunks of these files through the network on a by-need basis
Suggestions how to use OpenStreetMap data:

- perform queries on **meta-data** (such as uploaders, modification dates, etc.)
- work with actual **geographic data** (store, query, etc.)
- provide a distributed **tile server** for OSM data
Agenda, Formalities
<table>
<thead>
<tr>
<th>day</th>
<th>date</th>
<th>activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tue</td>
<td>Dec 18, 2018</td>
<td>presentation of project topics</td>
</tr>
<tr>
<td>Wed</td>
<td>Dec 19</td>
<td>students gather in groups of 3-5, group composition announced to instructor, remaining students assigned assignment of slots for concept presentation</td>
</tr>
<tr>
<td>Tue</td>
<td>Jan 8, 2019</td>
<td>concept presentation (attendance required)</td>
</tr>
<tr>
<td>Wed</td>
<td>Jan 9</td>
<td>concept presentation (attendance required)</td>
</tr>
<tr>
<td></td>
<td>Jan 15</td>
<td>assistance every Tue, Wed until Feb 6</td>
</tr>
<tr>
<td>Sun</td>
<td>Feb 10 23:59</td>
<td>submission of source code</td>
</tr>
<tr>
<td></td>
<td>Feb (day TBD)</td>
<td>oral exam (see later slide)</td>
</tr>
</tbody>
</table>

- **time and room** on the respective day same as lecture/tutorial
- concept presentation is prerequisite for oral exam
Discuss the following:

1. which *crates* will be used, and why:
   - how crate functions/types will be used
2. most important custom data *structures* and their purpose
3. how to solve the problem of *data/work distribution* among processes

**Presentation**

- length: about 8-10 slides
- duration: 15-20 min, then discussion
- slots available on January 8th and 9th, during lecture/tutorial hours
- slides to be handed in after the talk

**Purpose**

- stimulate discussion
- slide contents not binding for implementation
- initial presentation not relevant for grade
Implementation phase

- Implementation can be started before Jan 8, if desired.
- **Assistance** every Tuesday, Wednesday — during the time of lecture/tutorial.
- Please arrive at the **start of lecture** (10:15) or tutorial (14:15).
- An **email** in advance allows us to prepare for your question.
The final presentation constitutes the oral exam.

The final presentation consists of:

1. a description of the project, presented by one or more group members.
2. a code review where each group member must be able to point out his/her code contribution and discuss its implementation.

The final presentation has to be done in English.

Students have to bring hardware (i.e. laptops) in order to adequately present their projects, the network infrastructure will be provided by PMS in form of a WiFi access point.

Each group will be granted 45 minutes for their final presentation.