Exercises for

# Database Implementation

Elite Graduate Program Software Engineering

Florian Funke (florian.funke@in.tum.de)

## Assignment 1

### Info

- Due date: $15^{th}$ April 2014, 9:00am.
- Please use the prefix `[dbimpl]` in your submission email's subject and include information how to clone your `git` repository as well as the branch and commit ID.
- Please ensure your repository can be accessed by `florianfunke` (if you use Bitbucket) or `florian.funke@in.tum.de` (if you use GitHub).
- Please provide (minimal) documentation and comment your code.

### Excercise 1

Write a function `void externalSort(int fdInput, uint64_t size, int fdOutput, uint64_t memSize)` that sorts `size` 64 bit unsigned integer values stored in the file referred to by the file descriptor `fdInput` using `memSize` bytes of main memory and stores the result in the file associated with the file descriptor `fdOutput`. Your function should implement the external merge sort algorithm and should perform a $k$-way merge during the merge phase, i.e. merge $k$ runs together at once.

To sort individual runs, you may use STL's `std::sort` (from `<algorithm>`). To manage the $k$-way merge, the STL `std::priority_queue` (from `<queue>`) may be helpful.

### Excercise 2

Write a test case that sorts 5GB of data and verifies the order of the output. The command-line interface must be `sort <inputFile> <outputFile> <memoryBufferInMB>`. You'll find an input file generator on the class website that you may find useful for testing purposes. Your data format must adhere to the format specified in the program.

### Excercise 3 (Optional!)

Compare the performance of your implementation with an implementation that is agnostic of the memory hierarchy, e.g. one based on `std::sort` and `mmap`.

### Note

Literature on external merge sort:

- D. E. Knuth *The Art of Computer Programming, Volume III*: Replacement Selection
- J. S. Vitter *Algorithms and Data Structures for External Memory*: External Merge Sort[1]
- K. Mehlhorn and P. Sanders *Algorithms and Data Structures*

These systemcalls may be helpful:

- `open/close`
- `write/read` and `pwrite/pread`
- `posix_fallocate`
- ...

[1] Available Online: `http://www.ittc.ku.edu/~jsv/Papers/Vit.IO_book.pdf`