Project Seminar:
Parallel and Distributed Systems

Assignment 2 (Submission deadline: Nov 16th 2015, 23:59 CET)

General Rules

The assignment solutions have to be submitted at:

https://www.dcl.hpi.uni-potsdam.de/submit/

Our automated submission system is intended to give you feedback about the validity of your file upload. A submission is considered as accepted if the following rules are fulfilled:

- You did not miss the deadline.
- Your file upload can be decompressed with a zip / tar decompression tool.
- Your submitted solution contains only the source code files and a Makefile. Please leave out any Git / Mercurial repository clones or SVN / CVS meta-information.
- Your solution can be compiled using the "make" command, without entering a separate sub-directory after decompression.
- You program runs without expecting any kind of keyboard input or GUI interaction.
- Our assignment-specific validation script accepts your program output / generated files.

If something is wrong with your submission, you will be informed via email (console output, error code). Re-uploads of corrected solutions are possible until the deadline.

All tasks must be submitted accordingly in order to pass the assignment.

Assignment 2

The second assignment covers the study and implementation of parallel sorting algorithms as well as java monitors. You have to solve the given programming exercises in C/C++ with the POSIX PThread API[1]. Additional threading libraries are not allowed for this assignment; they will be covered in later assignments.

[1] `man pthread`

Task 2.1 Dining Philosophers with Java Monitors

Implement the dining philosophers with a freely chosen deadlock-free solution strategy.

Each philosopher has to be represented by a thread. You are free to map also other stake holders (e.g. waiters) or resources (e.g. forks) to threads if necessary.

Input

Your application has to be named "dinner" and accept two parameters, the number of philosophers resp. forks (min. 3) and the maximum run time in seconds.

```
Example: java -jar dinner.jar 3 10
```

Output

After the given run time is exceeded, the program must terminate with exit code 0 and produce an output file with the name of "output.txt" in the same directory. This file has to contain nothing but the number of "feedings" per philosopher as semicolon separated values.

```
Example: 5000;5000;5000
```

Task 2.2: Parallel sorting - theory

Choose a sorting algorithm and read about parallelization strategies for the chosen algorithm. For an overview of well parallelizable algorithms, have a look at: http://researcher.ibm.com/researcher/files/ie-albert_akhriev/sort2011-full.pdf .

Please submit a text file with your answers to the following questions, write a few (2-5) sentences to every questions:

1. Which algorithm did you chose and why?
2. How does it work?
3. What is the parallelization strategy?
4. What are worst cases?

Task 2.3 Parallel sorting - implementation

Implement your chosen algorithm of Task 2.2 using C/C++ and PThreads. For this exercise we provide you a framework to start with. It includes the files `main.cpp`, `sort.h` and `sort.cpp` . You are not allowed to alter `main.cpp` as your changes will be overwritten by the validation of submit. Please implement your chosen sorting algorithm using the function stub `your_sort_algorithm` in `sort.cpp`. You can alter the `sort.cpp` and `sort.h` as long as the function signature stays the same. You are also allowed to add more source and header files to the project. We included an example makefile. If you want to write your own or alter the provided one, keep in mind that the application has to be named "parsort".