Final Projects

No Exam . . .

This course doesn't have an exam.

But a Final Project!

Instead, the course concludes with a final project.

Start thinking about your projects now, but begin working in earnest in mid-December.

Projects due at the end of Week 2, 2018. (**January 12, 2018**)

Thus, you have lots of time for the project (plan carefully to actually get it done).
Each project should be roughly **one week** of work (per student).

Learning Outcomes

The project will exercise and assess your ability to

- write programs using different models and language abstractions for concurrent and parallel programming,
- analyze models with respect to which specific problems they address, where they can be used and where they should be avoided,
- present your results in writing.

Step 1: Find a Partner

Projects are assessed individually. However, you will need to *collaborate* with a partner for some parts of the project.

Please find yourself a project partner!

Deadline: Monday, November 13, **end** of the first lab

If you have not found a project partner by then, please inform George (`georgios.kalamatianos@it.uu.se`) right away.

If your group partner becomes unresponsive during the project or if there is a conflict that you cannot resolve quickly, **contact George or me** as soon as possible!

Step 2: Find a Topic

You are free to suggest your own project topic. Please talk to your project partner to find a topic that interests both of you!

Suitable topics require

- the implementation of a concurrent program, with
- about 3-4 days of programming work (for each person).

Step 2: Find a Topic

You are free to suggest your own project topic. Please talk to your project partner to find a topic that interests both of you!

Suitable topics require

- the implementation of a concurrent program, with
- about 3-4 days of programming work (for each person).

Some ideas:

- A rudimentary web server
- A simple image processor
- A simple chat tool
- A big data computation
- A machine learning algorithm
- A simple multi-agent system
- A simple computer-algebra system
- A multi-player game (engine)

Great topics require some communication between concurrent tasks, non-trivial shared state or some non-trivial way of distributing data. Look for interesting parallel algorithms!

Step 3: Choose a Concurrency Abstraction

Choose one of the concurrency abstractions covered in the course:

- Transactional Memory
- Actors
- Map-Reduce and/or Spark

You may also choose multiple abstractions and do a comparative project.

Choose an abstraction that will allow you to implement a solution to your project topic. For instance, map-reduce may not be appropriate for all topics.

Write a Project Description

When you and your partner have agreed on a topic, write a brief (ca. 200–300 words) **description of your project**.

Focus on a functional specification of the program that you want to implement: what is its purpose; what will it do? What will it *not* do?

Also include the **name of your project partner** and the **concurrency abstraction(s)** you have chosen.

The functional specification should mention all major features, but it does not need to be very detailed yet. You can refine it during the course of your project if necessary.

Submit Your Project Description

During the lab tomorrow, **discuss** your project description with George to make sure the scope of your project is reasonable.

Then submit your project description on the Student Portal.

Deadline: Monday, November 27, 18:00

Step 4: Design and Implementation

Implement your program, using the chosen concurrency abstraction(s).

You may use any programming language(s). Of course, the language(s) that you use must provide the concurrency abstraction that you chose earlier. If you are not sure about this, talk to George or me.

Make sure that your program can be compiled and run on the lab machines.

If this is not possible—for instance, if you want to use a language for which there is no compiler available on the lab machines—talk to George or me first.

Step 5: Write a Report — Individual

Write a **report** (ca. 1,000–1,500 words). The report must include:

- Title, your name, your partner's name, table of contents
- Introduction: a summary of what the program does
- Use cases: how to compile and run your program, including key examples
- Program documentation: a description of important data structures, algorithms, functions
- Performance evaluation: a discussion of your program's performance, e.g., observed speedup on multi-core hardware
- Concurrency abstractions: a discussion of the benefits and drawbacks of your chosen concurrency abstraction in the context of your project, especially in comparison to your partner's abstraction
- Known shortcomings: a description of things that don't work as well as you would like, or that could be added in the future
- What you did: a description of which parts of the project you did.

Reading the report should be sufficient to understand your program!

Your report should be self-contained. It is okay if there is some overlap with your partner's report, but all discussion sections should be your own ideas in your own words.

Submit Your Code and Report

Submit your source code **and** your report (as a PDF file) on the Student Portal.

Every student needs to make a submission to Studentportalen.

Deadline: Friday, January 12, 2018, 18:00

Assessment

You need to pass the project to pass the course. The written **report** (including your source code) is worth 20 points.

Grading will be based on:

- the *presentation* of your material (e.g., spelling, readability, use of references, figures and visual aids),
- your *effort* (amount of work done, difficulty of the project),
- your *reasoning* (e.g., quality of explanations, apparent level of understanding, code quality).

Good luck!