AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

Oral exam in I3ISU

Søren Hansen <shan@iha.dk>

January 14, 2018

Version 1.5

Preamble

This is a small folder describing what will be expected of students participating in the I3ISU oral exam.

Expectations & Process

The exam is a classical oral exam in the sense that you get to pick a paper, amongst a number of papers, which then tells which subject you will be examed in.

At this point you will be expected to start elaborating upon this paricular subject, both from a theoretical point of view as well as a practical point of view. This is where the exercise solutions and the associated wikis that you have made during the course come in handy. These can be used as an aid throughout the exam by serving as input for further discussion.

However this is *not* a guarantee that you will not be presented with code snippets[1] that you have not seen before. Is this the case, you must be adequately versed in the curriculum to discuss the various principles, concepts and challenges. This includes being able to relate and put input perspective, the different concepts within a subject as well as between subjects.

Valid aid: an outline and lab solutions.
Note that computers are not accepted. This thus means that no slideshow is viable.

The examination takes approximately 15 minutes, after which the examinee leaves. Upon reaching an agreement on the grade the examinee will be asked to enter and the grade will be presented.

[1]If relevant for the topic, certain function and their signatures might be important as well. It says code, however UML diagrams are just as likely

Subjects

In the follow the different subjects that the oral exam comprises of will be shown, and some sub topics that illustrate the particular subject have been added for improved understanding of the particular subject. This is followed by the corresponding curriculum as well as which exercises are deemed relevant for this particular subject.

Subject	Sub Topics	Curriculum	Exercises
Programs in relation to the OS and Kernel	• Processes and threads • Threading Model • Process anatomy • Virtual Memory • Threads being executed on CPU, the associated scheduler & Cache	• Slides “Intro to OSs” • Slides “Parallel Programs, Processes and Threads” • OLA: “Anatomy of a Program in Memory” by Gustavo Duarte • OLA: “The Free Lunch is Over” • OLA: “Virtual Memory :p131-141(until AVL trees)” • OLA: “Introduction to Operating Systems” • OLA: “Multithreading” • Kerrisk: “Chapter 3-3.4: System Programming Concepts” • Kerrisk: “Chapter 29: Threads: Introduction”	Posix Threads
Synchronization and protection	• Data integrity - Concurrency challenge • Mutex & Semaphore • Mutex & Conditionals • Producer / Consumer problem • Dinning Philosophers • Dead locks	• Slides “Thread Synchronization I & II” • Kerrisk: “Chapter 30: Thread Synchronization” • Kerrisk: “Chapter 53: Posix Semaphores (Named not in focus for this exercise)” • OLA: “pthread-Tutorial - chapters 4-6” . • OLA: “Producer / Consumer problem” • OLA “Dining Philosophers problem” • OLA “How to use priority inheritance”	Posix Threads & Thread Synchronization I & II

Subject	Sub Topics	Curriculum	Exercises
Thread Communication	• The challenges performing intra-process communication • Message queue – The premises for designing it – Various design solutions - Which one chosen and why – Its design and implementation • Impact on design/implementation between before and after the Message Queue • Event Driven Programming – Basic idea – Reactiveness – Design - e.g. from sequence diagrams to code (or vice versa)	• Slides “Inter-Thread Communication” • OLA: “Event Driven Programming: Introduction, Tutorial, History - Pages 1-19 & 30-51” • OLA: “Programming with Threads - chapters 4 & 6”	Thread Communication
OS Api	• The design philosophy - Why OO and OS Api? • Elaborate on the challenge of building it and its current design – The PIMPL / Cheshire Cat idiom - The how and why – CPU / OS Architecture • Effect on design/implementation – MQs (Message queues) used with pthreads contra MQ used in OO OS Api. – RAII in use – Using Threads before and now • UML Diagrams to implementation (class and sequence) - How?	• Slides: “OS Api” • OLA: “OSAL SERNA SAC10”. • OLA: “Specification of an OS Api” • Kerrisk: “Chapter 35: Process Priorities and Scheduling”	OS API

Subject	Sub Topics	Curriculum	Exercises
Message Distribution System (MDS)	• Messaging distribution system - Why & how? • The PostOffice design - Why and how? • Decoupling achieved • Design considerations & implementation • Patterns per design and in relation to the MDS and PostOffice design – GoF Singleton Pattern – GoF Observer Pattern – GoF Mediator Pattern	• Slides: “A message system” • OLA: “GoF Singleton pattern” • OLA: “GoF Observer pattern” • OLA: “GoF Mediator pattern”	The Message Distribution System
Resource handling	• RAII - What and why? • Copy construction and the assignment operator • What is the concept behind a *Counted SmartPointer*? • What is `boost::shared_ptr<>` and how do you use it?	• Slides: “Resource Handling” • OLA: “RAII - Resource Acquisition Is Initialization” • OLA: “SmartPointer” • OLA: “Counted Body” • OLA: “boost::shared_ptr” • OLA: “Rule of 3”	Resource Handling