

## Introduction

So far the basic or fundamental building blocks that need to be utilised for thread synchronisation and communication have been examined and used. Through this it became somewhat evident that this approach, based on the fundamentals, have some rather challenging side effects. As a consequence of this we will in this lecture try to investigate another paradigm namely the *Event Driven Programming* paradigm. This is no *Silver Bullet*<sup>1</sup> and presents its own design and implementational issues.

## Content and reflection

### Themes

- Event Driven Programming (EDP)[3, pages 1-19 & 30-51][1]
- Thread Synchronisation
  - Protection and Synchronization[5, chap. 30]
  - Posix semaphores[5, chap. 53][2]  
*Excluding named*
- Buffer case - Producer/Consumer[4][2]  
*Implemented using semaphores*

### Questions

- Event Driven Programming (EDP)
  - EDP is *reactive programming* - what does this entail
  - What kind of work does an EDP system do - applies to many system. One could think about a *Blue-Ray player*...
- Event
  - What *is* an event / signal / message
    - \* From a conceptual point of view
    - \* From an implementational point of view
  - What does it mean to *passing an event from on thread to another*
  - What does the term event handler cover and what does it actually do
  - The difference in what a *dispatcher* is and what an *event handler* is. Note that the literature makes a distinction between these two terms, whereas the presentation just calls them the *event handler*
  - What is an event loop
  - Who should allocate a given event and who should free it

<sup>1</sup>Silver Bullet is a reference to a solution that solves everything. No such one exists

## NOTE: Refreshing / Presumed known

- Sequence and State diagrams
- Inheritance - idea and notation in C++
- What is a *downcast* in relation to inheritance
- What does RTTI and `typeid()` provide

## Material

### Slides

- [1] S. Hansen, *Thread communication*, Slides - see course repos.
- [2] —, *Thread synchronization i*, Slides - see course repos.

### Local repository

- [3] S. Ferg, *Event-driven programming: Introduction, tutorial, history*, Tutorial, version 0.2, Feb. 2006.

### Online

- [4] E. al. (). Producer–consumer problem. Wikipedia Article, [Online]. Available: [https://en.wikipedia.org/wiki/Producer%2dconsumer\\_problem](https://en.wikipedia.org/wiki/Producer%2dconsumer_problem).

### Hardback

- [5] M. Kerrisk, *The Linux Programming Interface*. No Starch Press, Inc, 2010, ISBN: 978-1-59327-220-3.