OS API

Søren Hansen <sha@ase.au.dk>
V1.0

Introduction

Designing and implementing code that is to run on a number of different platforms is a challenge since this obviously requires in-depth knowledge on each. In order to decouple knowledge for a given platform from the application to be developed, an extra layer is created and injected. This layer is the so-called OSAL (OS Abstraction Layer). In this lecture we are going to get to grasps with what this actually is, how it works, how to use it and to some extend how it is implemented. As a bonus we will use an Object Oriented approach thus giving us an OO OSAL.

Content and reflection

Themes

- An OSAL case for Portable apps in wireless sensors networks[2]
- Specification of an OS Api[3]
  Get a feel for what specification can look like and whats expected of the OS API
- Using the OS Api in an application[4]
  Note that the eclipse part is irrelevant
- APIs, frameworks, design to implementation[1]
- Idioms/Patterns
  - Strategy pattern[5][1]
  - pimpl idiom[6]
- Other
  - GCC compiler specifics[7]

Questions

- Basics
  - What is an API
  - What is a framework
- OSAL - Our OS API
  - What does this concept cover
  - Why would we need it
  - How is it structured (see files) (header, cpp files, makefiles...)
  - What does "*hiding implementation from header files*" e.g. pimpl idiom. Where can this be sceen in the OS API implementation
  - How do I create an *active thread* using our new OS API (inheritance, method implementation ...)
  - Where do I *put* the events a given *active thread* understands
  - Each thread is supposed to have its own message queue, how does this fit in with the OS API threads

– In which method is the event loop supposed to be implemented

Material

Slides

[1] S. Hansen, *Os api*, Slides - see course repos.

Local repository

[2] I. S.G. F. Ramon Serna Oliver, "An operating system abstraction layer for portable applications in wireless sensor networks", 2010, Filename: OSAL SERNA_SAC10.pdf.

[3] S. Hansen, *Specification of an os api*, version 3.3, 2014.

[4] ——, *Using the new osapi in eclipse and makefiles*, version 1.0, 2012.

Online

[5] Unknown. (2016). Strategy pattern, [Online]. Available: `https://en.wikipedia.org/wiki/Strategy_pattern`.

[6] ——, (2016). Pointer to implementation (pimpl), [Online]. Available: `https://en.wikibooks.org/wiki/C%2B%2B_Programming/Idioms#Pointer_To_Implementation_.28pImpl.29`.

[7] B. J. Gough. (). Link order of libraries when using gcc, [Online]. Available: `http://www.network-theory.co.uk/docs/gccintro/gccintro_18.html`.

Remember thy makefiles, otherwise ...