FACULTY OF MECHATRONICS,
INFORMATICS AND INTERDISCIPLINARY
STUDIES TUL

Learn the OS theory while working with Zephyr

Lenka Kosková Třísková
Technical University of Liberec

lenka.koskova.triskova@tul.cz

Who am I

- I was always more an engineer than a scientist.
- As a freshly graduated DSP engineer, with no glue of the OS theory, I was responsible for writing OS kernel in DSP assembler 20 years ago.
- It was a nightmare.
- Now I teach operating systems and embedded systems at Technical university of Liberec.
- I'm trying to teach what would have been helpful for me to know at the time.

Who are my students

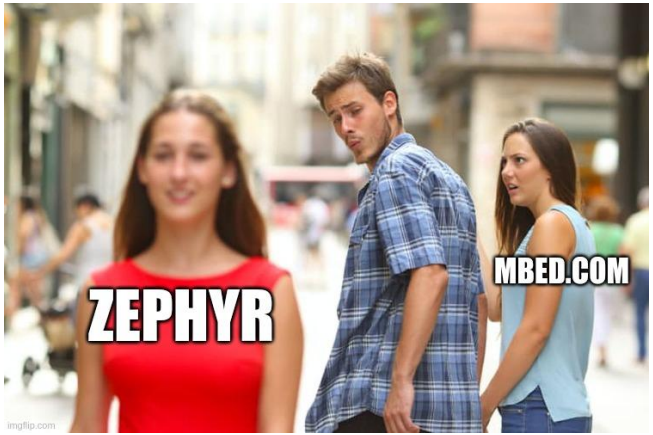
- They are in 2nd year of study
- Not so experienced in C/C++
- Very limited HW knowledge
- Half of them came from “general” high school
- The rest comes from “industry” high schools focused on electronics

How we teach

- 14 Theoretical lectures (90 minutes each) for all the students
- 14 Practical labs (90 minutes each) for groups of 20-25 students
- The labs shall fix what was explained at theoretical lectures
- In Czech, labs are mandatory, lectures are optional

How to select a model OS for the labs

- We need a showcase OS!
- Linux is too complex
- RTOS is more suitable
- 2016 mbed.com
  - Pros: online, wide support for devices, open source, handy examples
  - Cons: HW we use is not supported by ST (no updates for examples and docs)
  - Cons: the online IDE changed in 2022 and starts to be slow and confusing for students
  - Cons: other promising systems on the scene!

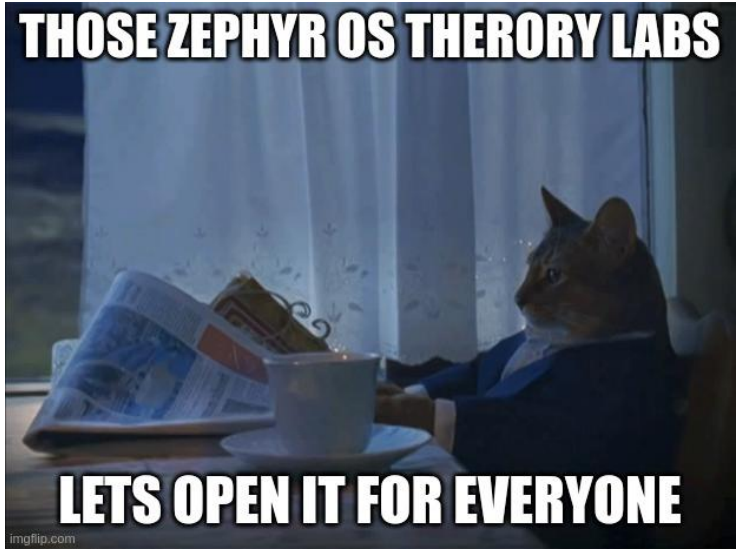


Why Zephyr?

- It is cool, modern, rising star in RTOS world
- Wide HW support
- Emulation with qemu
- Documentation and community
- Libraries
- Reliability
- Device trees

The zephyr-os-labs

- Available on gitlab: https://gitlab.com/lenkakt-git/zephyr/zephyr-os-labs
- Template lab
- General Readme
- Each lab in separate folder as an application
- Instructions in the Readme
- In src, there is always a solution directory

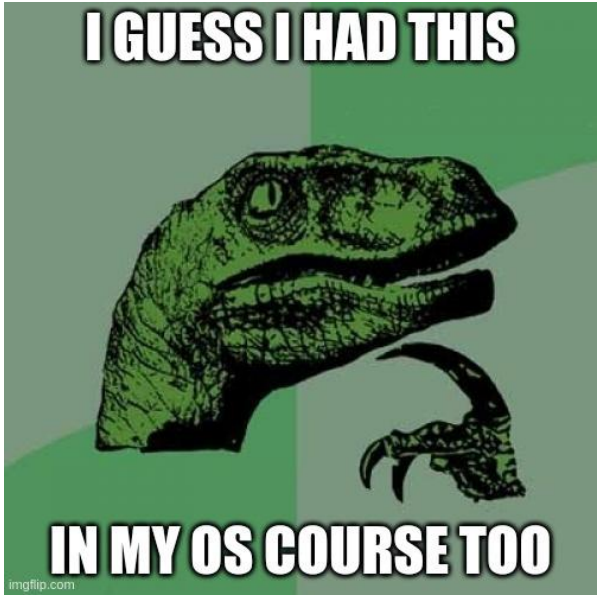


Still under construction

- Each lab has defined content
- Slowly implementing, hard deadline 09/2024
- We are using qemu (arm, x86) and Hardwario Chester platform

The class structure - talks

OS features in general
HW overview
Kernel concepts
Processes and threads
Scheduling
Memory management
Synchronization
Device management
Storages and filesystems
RTOS and real time
Standards
Security

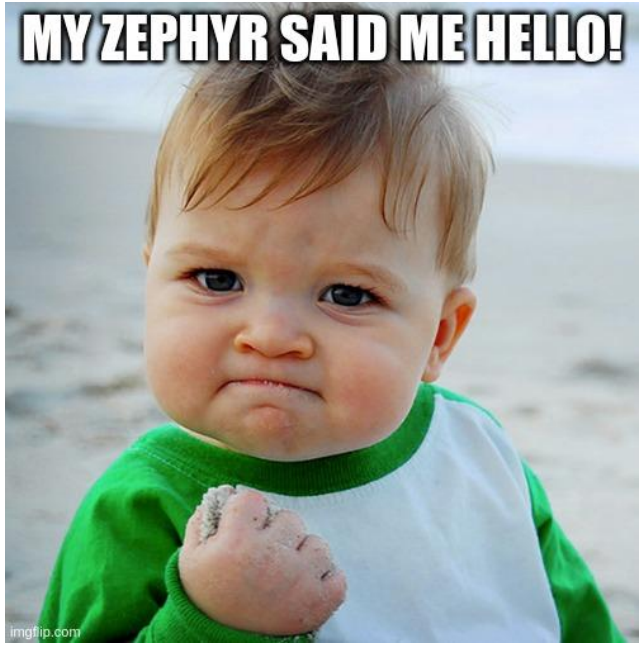


The talks and labs

OS features in general	“hello” - Get it working
HW overview	“config” - Get to know the configuration
Kernel concepts	“kernel_lab” - Get familier with outputs, API demo
Processes and threads	“thread” - Creating threads, user-space threads
Scheduling	“scheduling” - Scheduler options, priorities, preemption
Memory management	“memory” - Heaps and slabs, virtual memory
Synchronization	“synchro” - Locks and semaphores
Device management	“device_tree” - Device trees and drivers
Storages and filesystems	“filesystems” - FS support
RTOS and real time	
Standards	“posix” - Working with posix subsystem
Security	

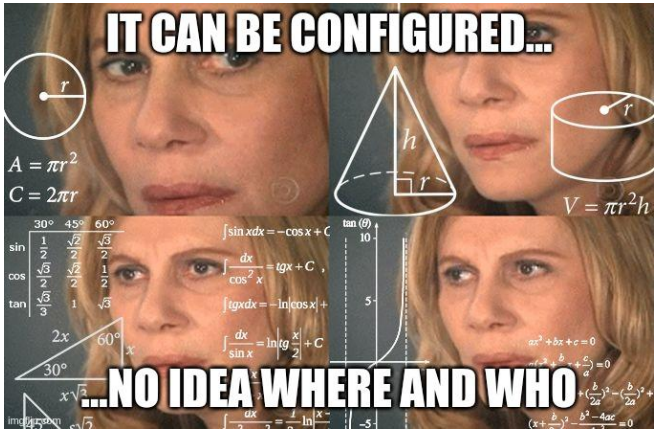
“Hello” - Get it working

- Goal: Install the Zephyr and compile the “Hello world”.
- The west tool
- The directory structure
- The qemu
- Status: approved by students, working



“config” - CMake, KConfig, command line

- Goal: Where are all the inputs affecting my output?
- CMake
- KConfig fragments
- Menuconfig
- Parameters from command line
- Status: Done, approved by students.



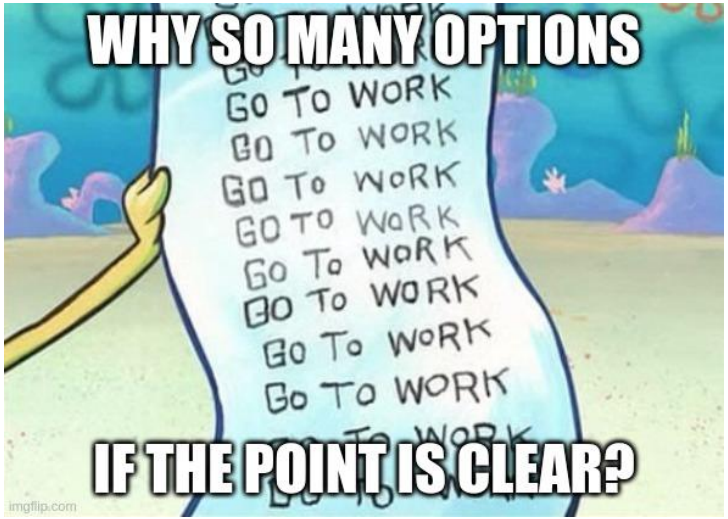
“threads” - Thread (process concept)

- Goal: Learn how to spawn a thread and understand a thread elementary parts
- What I need to start a thread (stack, entry function etc.)
- Macros to start threads
- Macros parameters
- Kernel threads vs. user-space threads
- Thread features
- Status: Done and approved by students.

“scheduling” - Play with the scheduler

Goal: Scheduler algorithms, priority and preemptive scheduling

Status: 1st lab implemented and approved by students.

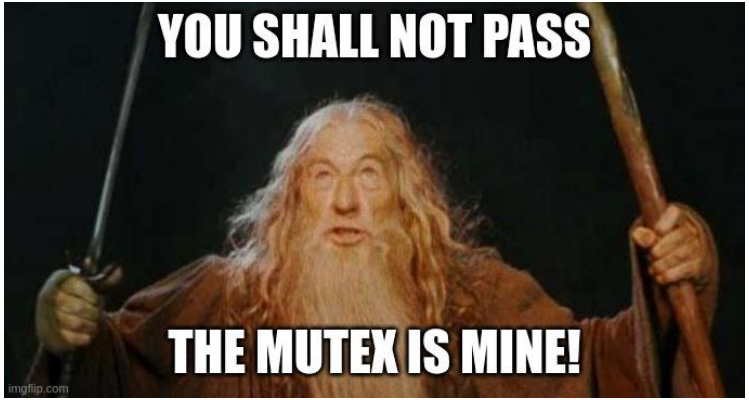


“synchro” - Mutexes and semaphores

Goal: Understand the elementary thread synchronization

Threads are using mutex when they try to write into the serial line

“Parking lot” - threads simulate cars trying to access the parking lot driven by a semaphore.



“memory” - Heaps, slabs, virtual memory

Goal: Learn, how the OS deals with memory

Simple heap usage

slab example (with a short view to Linux)

virtual memory example

“filesystems” - Play with the FAT subsystem

The goal: To demonstrate what is going on when you call an “open” function from any fs API

Create a small FAT fs, emulate SD card and access files from a thread