Course Prospectus

These are the topics we plan to cover, and the time we plan to spend covering them. This is all very flexible, and will likely change rapidly.

Times in parens are hours and minutes for the topic. `[day 1]` etc are course days. Links are to associated material.

Course Days

1. Wednesday 8 January
2. Friday 10 January
3. Monday 13 January
4. Wednesday 15 January
5. Friday 17 January
6. Monday 20 January
7. Wednesday 22 January
8. Friday 24 January

Schedule (20:55)

- Rust review (10:30)
    - Basic Datatypes [day 1] (2:50)
        - Numbers [numbers] (:15)
        - Arrays and slices [arrays and slices] (:40)
        - Strings and chars [strings] (:30)
            - Unicode, UTF-8, UTF-16 (:15)
            - slices (:15)
        - Structs and enums structs and enums (:55)
            - Structs (:15)
            - Enums (:20)
            - Method syntax and semantics (:20)
        - Exercise: Rule 110 (:30)
    - Ownership, lifetimes and borrowing [day 2] (2:25)
        - Rust memory model [memory and moves] (:15)
        - Move semantics [memory and moves] (:15)
        - Memory management (:55)
            - `Box`, mutability [smart pointers] (:10)
            - `RefCell` and `Mutex` [interior mutability] (:30)
            - `Rc` and `Arc` [smart pointers] (:15)
        - Globals, `'static`, leaking (:30)
        - Live Coding: Histograms (:30)
    - Rust types and generics [day 3] (2:35)
        - Generics (1:00)
            - Basic generics [generics] (:20)
            - Type aliases [typedef] (:10)

            - Const generics (:10)
            - Generic lifetimes (:20)
        - Pattern Matching [patterns] (:20)
        - Traits [traits] (:45)
            - Trait basics (:20)
            - Associated types (:15)
            - Trait inheritance (:10)
        - Exercise (:30)
    - Finishing the basics [day 4] (2:30)
        - Catch-up and Review (1:15)
        - Error Handling [errors] (1:15)

- Rust software development [day 5] (2:55)
    - Code Checking, Testing, Benchmarking, Debugging [testing] (:15)
    - Crates and modules [modules] (:15)
    - Libraries (:55)
        - `std`, `core` and `alloc` (:15)
        - Cargo and `crates.io` (:15)
        - A tour of `std` and `crates.io` (:15)
        - Cargo `features` (:10)
    - Exercise: Rust project code reading (1:00)

- "Advanced" Rust [day 6] (2:35)
    - Flexible programming (1:10)
        - Fancy traits (:30)
            - Trait objects [trait objects] (:15)
            - Deref and related traits [deref] (:15)
        - Closures and iterators (:40)
            - Closures [closures] (:20)
            - Iterators [iterators] (:20)
    - Unsafe and `no_std` (:55)
        - Unsafe fundamentals [unsafe] (:40)
            - Basics, safety docs (:30)
            - Nightly (:10)
        - `nostd` (:15)
    - Exercise: Real-time scheduler (:30)

- Embedded Programming [day 7] (2:45)
    - Bare Metal Programming (1:15)
        - PAC (:15)
        - HAL (:20)
        - Peripheral sharing (:20)
            - `Arc, Mutex` (:10)
            - `PeripheralRef` [peripheral_ref] (:10)
        - FFI [ffi] (:20)
    - Async/await [async-await] [day 8] (1:00)

        - Top-half: user view (:30)
        - Embassy [embassy] (:30)
    - Exercise (:30)

- Rust Design Patterns [day 8] (2:15)
    - Functional programming (:30)
    - Trait Sealing [trait sealing] (:15)
    - Marker Traits (:30)
        - `Copy` (:05)
        - `Sized` (:05)
        - `Send` and `Sync` (:20)
    - Typestate Driven Programming [typestate] (:30)
    - Exercise (:30)
