Week 08 Workshop

EXERCISE:
Workshop 8 - Async Scheduling

PRE-WORKSHOP PREPARATION:

This week, the only preparation is to watch the lectures, and read over the starter code.

IN-WORKSHOP REVISION:

Your tutor will revise the following concepts briefly:

- Threading (spawn and scope)
- Sync and Locking primitives (Arc, Mutex, etc.)
- The Send and Sync traits

One of the most powerful reasons Rust is amazing is because it is easy to drop in concurrency into an existing system. Say, for example, you have a `Vec`. If you're doing operations that don't need to be done in order, and don't need to know other elements, you can easy change your code to use paralellism to speed up your code.

Before starting, you should download the starter code, here.

In this lab, you'll be writing a program called `viscose`, which will act similarly to the `rayon` library. It will take in a vector, and perform one of three operations on that vector. The three operations are:

- find: find the first index of the element with the given value.
- find_all: find every index of an element with the given value.
- map: run a function over every value in the list.

In this lab, you'll be working through the following tasks:

Task 1

Implement the `find` function, such that it takes an `i32`, and returns the first index of the vec that has a matching number, or -1 otherwise. Ensure that you can configure the number of threads that your program uses.

HINT:

The documentation for the `chunks` function may be useful.

Task 2

Once you've implemented find, test how large the vec has to be before it's faster to use the parallel version compared to the inbuilt version. Try and figure out what the optimal number of threads to use is before it slows down again. Try and explain your results (and see if they're the same on somebody else's computer).

Task 3

Implement the `find_all` function, such that it takes a number, and returns all indexes that have the matching number.

Task 4

Convert your code such that it works for a vector of any type (not just i32). You will need to use generics, and to ensure some of your generics implement the appropriate traits.

Task 5

Implement the `map` function such that you can write a closure which maps from a `T` to another `T`, and apply that to every element of the vector in parallel.

Extension 1

Implement the `map` function such that you can write a closure which maps from a `T` to another `U`, and apply that to every element of the vector.

Extension 2

Try and improve the performance of your functions. You might want to look into `split_at_mut`, and with_capacity as two ways to speed up the code.

Extension 3

Compare the speed of your functions to rayon's functions.

End of Class Discussion

- How easy/hard was it to get Rust to compile?
- Did you run into any concurrency issues while testing your code?
- What's one thing you learned today?
