# Actores

Técnicas de Programación Concurrente

# Un ejemplo de condvars en la vida real

Servo un rendering engine de browser implementado en Rust.

fetch() es una API web candidata a ser el nuevo standard para realizar requests HTTP desde JS

La implementación de cache de fetch de Servo sufría de una race condition cuando concurrentemente se realizaban dos requests al mismo recurso.

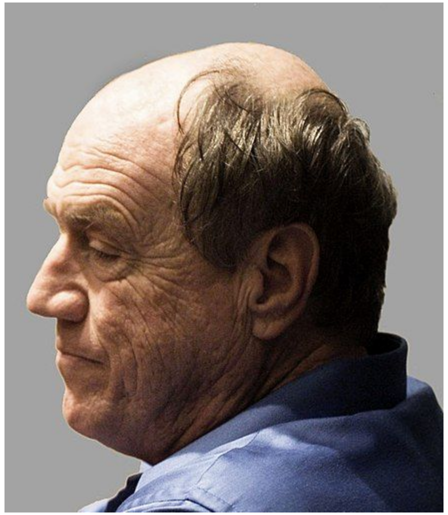
Lo solucionaron con una condvar

https://github.com/servo/servo/pull/24318/files#diff-5b7cef56895a34b09812a744ac8376802d4c38b76115148d8f9e3fc3458ca42eR84

# Actores

Vamos con unos ejemplos introductorios con Actix

- Hola mundo!
- Actores con estado interno
- Uso de sleep



Carl Hewitt
Paper original de **1973!**

## Problema del banquero

Solucionemos el problema de banquero utilizando actores

- Qué actores necesitamos?
- Qué tipos de mensaje?

## Problema de los Filosofos

Solucionemos el problema de los filósofos nuevamente, utilizando actores y la solución de Chandy/Misra

- Each fork can either be *dirty* or *clean.* Initially, all forks are dirty.
- When a philosopher wants to use a set of resources (*i.e.*, eat), said philosopher must obtain the forks from their contending neighbors. For all such forks the philosopher does not have, they send a request message.
- When a philosopher with a fork receives a request message, they keep the fork if it is clean, but give it up when it is dirty. If the philosopher sends the fork over, they clean the fork before doing so.
- After a philosopher is done eating, all their forks become dirty. If another philosopher had previously requested one of the forks, the philosopher that has just finished eating cleans the fork and sends it.

## Referencias

https://www.ijcai.org/Proceedings/73/Papers/027B.pdf

https://www.cs.utexas.edu/users/misra/scannedPdf.dir/DrinkingPhil.pdf