Message passing variability:

- Erlang - asynchronous message passing
  - per-process mailboxes

Asynchronous means senders don't wait when sending.

For receiving:
- receive is synchronous, even in Erlang.

- Many other combinations are possible:
  - Synchronous sends.

- Channels instead of mailboxes:
  - Channel is itself a value in the language.
  - A process may receive or send to different channels at different times.
A bit of history:

- Channels & synchronous MP.

Ada - late 1970s: Rendezvous -
- multi-way synchronous
- send & receive mechanism.

- OCCAM - based on CSP - Transputer
- Multi-processor - mid '80s

Theory:

\{ Calculus of Communicating Systems (CCS) \}
- Robin Milner

\{ Pi Calculus
- Process Calculus \}
Now we'll look at a real language using synchronous message passing and channels.

**Language** Concurrent ML

- based on ML language (Milner)
- extensions by John Reppy.

Statically typed, mostly functional language.

(Contrast Erlang: dynamically typed, mostly functional)

Concurrent ML:

- typed channels: each channel can only carry values of one particular type.

- type language
  - int channel
  - string channel
  - bool channel

- int list
- string list
- bool list
and thus we have operations to send and receive on channels:

\[ \text{Send} \left( c, v \right) : \quad \alpha \text{ channel} \times \alpha \rightarrow \text{unit} \]

\[ \text{recv} \left( c \right) : \quad \alpha \text{ channel} \rightarrow \alpha \]

Synchronous