
University of
Kent

Joe Armstrong

EXPERT SYSTEM DEVELOPER, AND ONE OF THE
CREATORS OF ERLANG AT ERICSSON

University of Kent

University of Kent

```
-module(counter0).
-export([start/0, loop/1, tick/1, read/0]).

start() ->
   register(counter0, spawn(counter0, loop, [0])).

tick(N) -> rpc({tick, N}).
read()  -> rpc(read).

loop(State) ->
   receive
     {From, Tag, {tick, N}} ->
       From ! {Tag, ack},
       loop(State + N);
     {From, Tag, read} ->
       From ! {Tag, State},
       loop(State)
   end.
```

```
rpc(Query) ->
   Tag = make_ref(),
   counter0 ! {self(), Tag, Query},
   receive
     {Tag, Reply} ->
        Reply
   end.
```

University of Kent

```
-module(gen_server_lite).
-export([start/2, loop/2, rpc/2]).

start(Mod, State) ->
   register(Mod, spawn(gen_server_lite, loop, [Mod,State])).

loop(Mod, State) ->
   receive
     {From, Tag, Query} ->
        {Reply, State1} = Mod:handle(Query, State),
        From ! {Tag, Reply},
        loop(Mod, State1)
   end.

rpc() ->
  ... as before ...
```

```
-module(counter1).
-import(gen_server_lite,[start/2,rpc/2]).
-export(...).

start() -> start(counter1, 0).

tick(N) -> rpc(counter1, {tick, N}).
read()  -> rpc(counter1, read).

handle({tick,N}, State) -> {ack, State+N};
Handle(read, State)     -> {State, State};
```

Summary

University of Kent

- Four primitives for concurrency (`spawn`, `send`, `receive`, `self`)
- Timeouts
- `register` – `whereis`
- Trapping errors
- How to build our own concurrency abstractions


University of
Kent