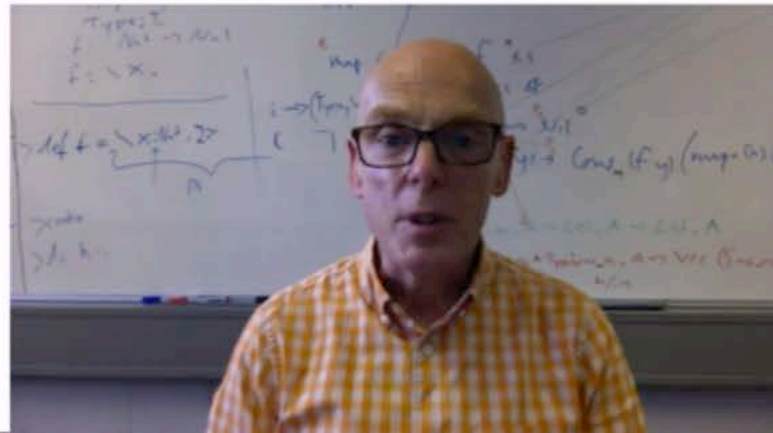

University of
Kent

University of Kent

Bulk storage



University of Kent

Emulating state in Erlang

We can use the constructs of Erlang to emulate mutable state.

We're familiar with a **tail recursive loop**, where the parameters represent the state.

```
loop(State) ->
  Process  = …,
  NewState = …,
  loop(NewState).
```

We can also do something similar with **messages sent to ourselves**: the succession of messages represents the history of the state values.

```
self() ! StartState,
loop() ->
  receive State ->
    Process  = …,
    NewState = …,
    self() ! NewState,
    loop().
```

University of Kent

But what if we want *real* state?

Suppose we want to keep track of substantial amounts of data in a program … recording and updating it as we compute.

Erlang Term Storage ETS and the persistent file-based DETS are libraries for tuple-based table storage.

```
{ant,1,2,3}
{bee,1,2,3}
{cat,1,23}
{cat,1,23,4456}
{dog,collie,342}
{eel,14456}
{flea,1,23,4456}
```

University of Kent

Different flavours: key on one field

Hash-tables

Sets: no repeated keys.

Bags: repeated keys ok, but no repeated tuples.

Duplicate bags: repeated keys and tuples both ok.

Trees

Ordered sets: sets + key order.

```
{ant,1,2,3}
{bee,1,2,3}
{cat,1,23}
{cat,1,23,4456}
{dog,collie,342}
{eel,14456}
{flea,1,23,4456}
```

University of Kent

Functionality

Lookup

Update

Search

Bulk operations

Transactionality

```
{ant,1,2,3}
{bee,1,2,3}
{cat,1,23}
{cat,1,23,4456}
{dog,collie,342}
{eel,14456}
{flea,1,23,4456}
```

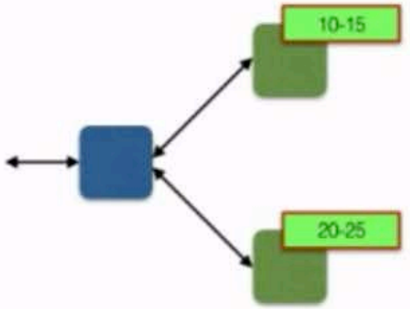
University of Kent

Where next?

Full details of ETS and DETS are given in the online manual pages

`http://erlang.org/doc/man/ets.html`

`http://erlang.org/doc/man/dets.htm`



As an exercise it would be possible to dump the state of a frequency server into a (D)ETS table, and reload from that in case of failure.


University of
Kent