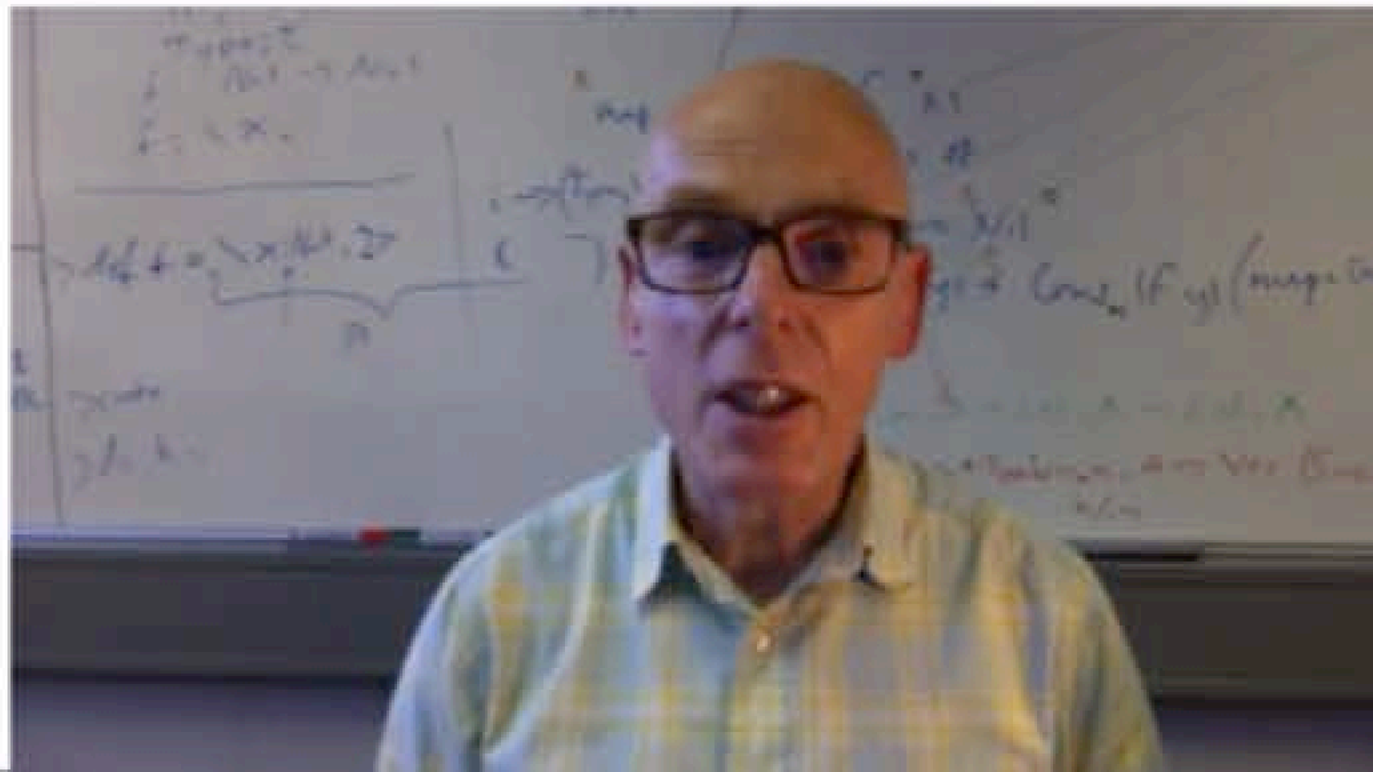
University of Kent

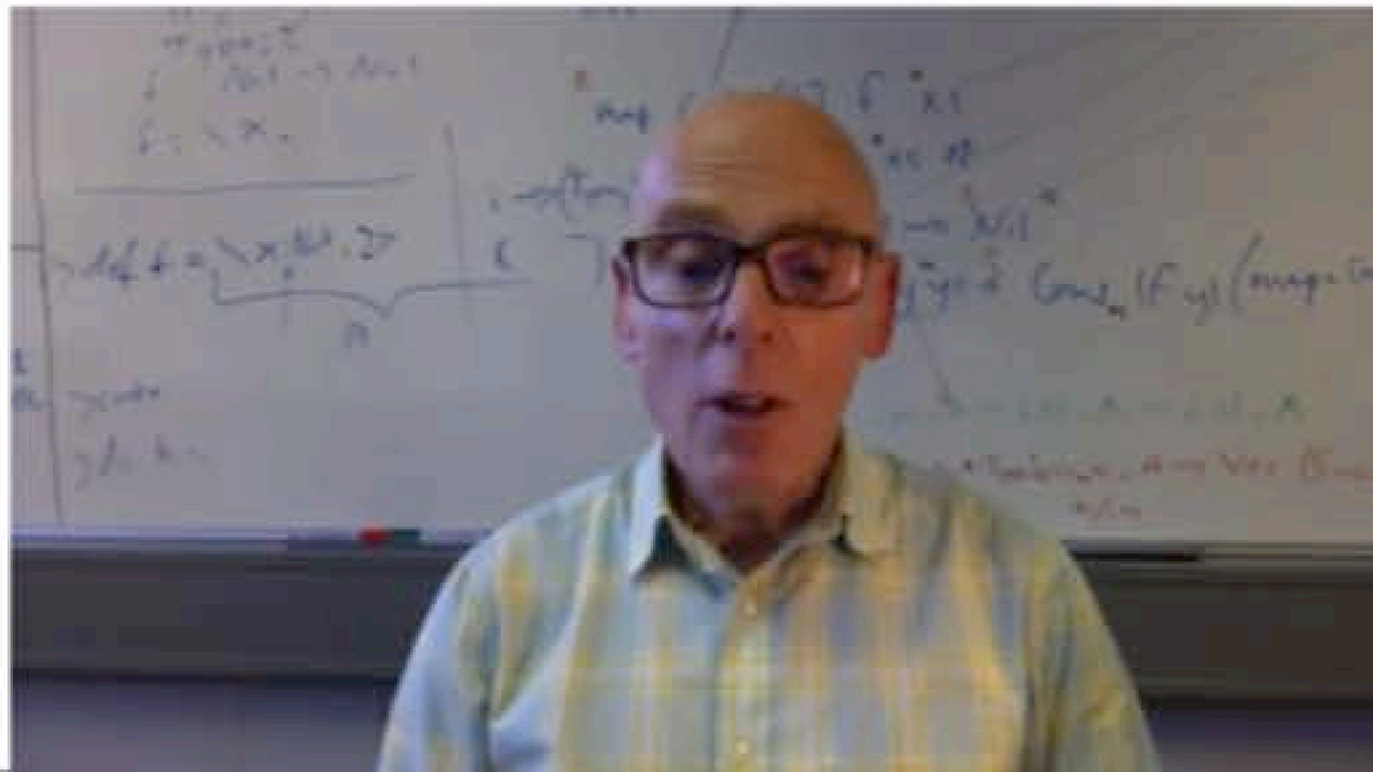
Press esc to exit full screen

Let it fail!



Process lifetimes

How long does a process "live" in Erlang?



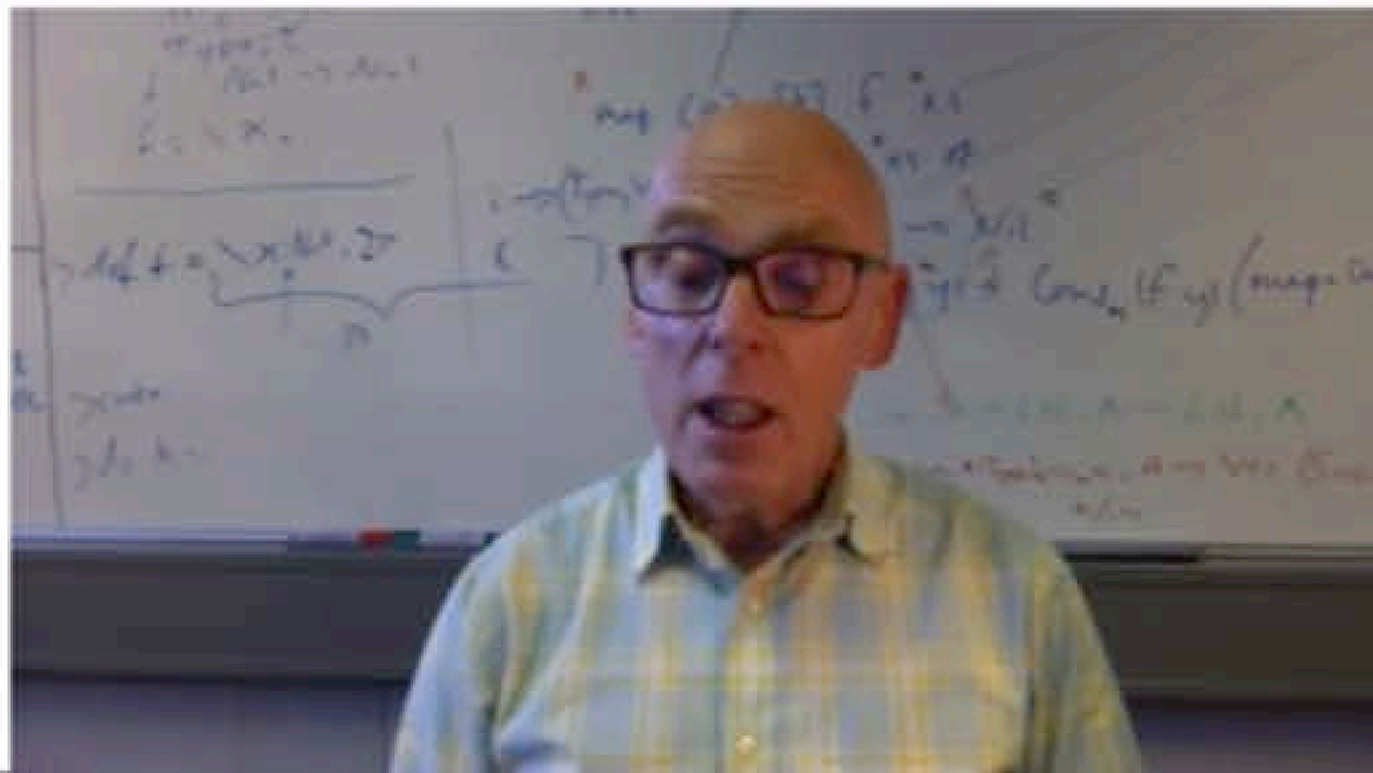
Processes can execute indefinitely

This process will `loop` forever …

… and acknowledge some messages via `io`.

```
spawn(?MODULE,loop,[]).

loop() ->
  receive
    {msg,M} ->
      io:format("ack"),
      loop();
    _Msg ->
      loop()
end.
```



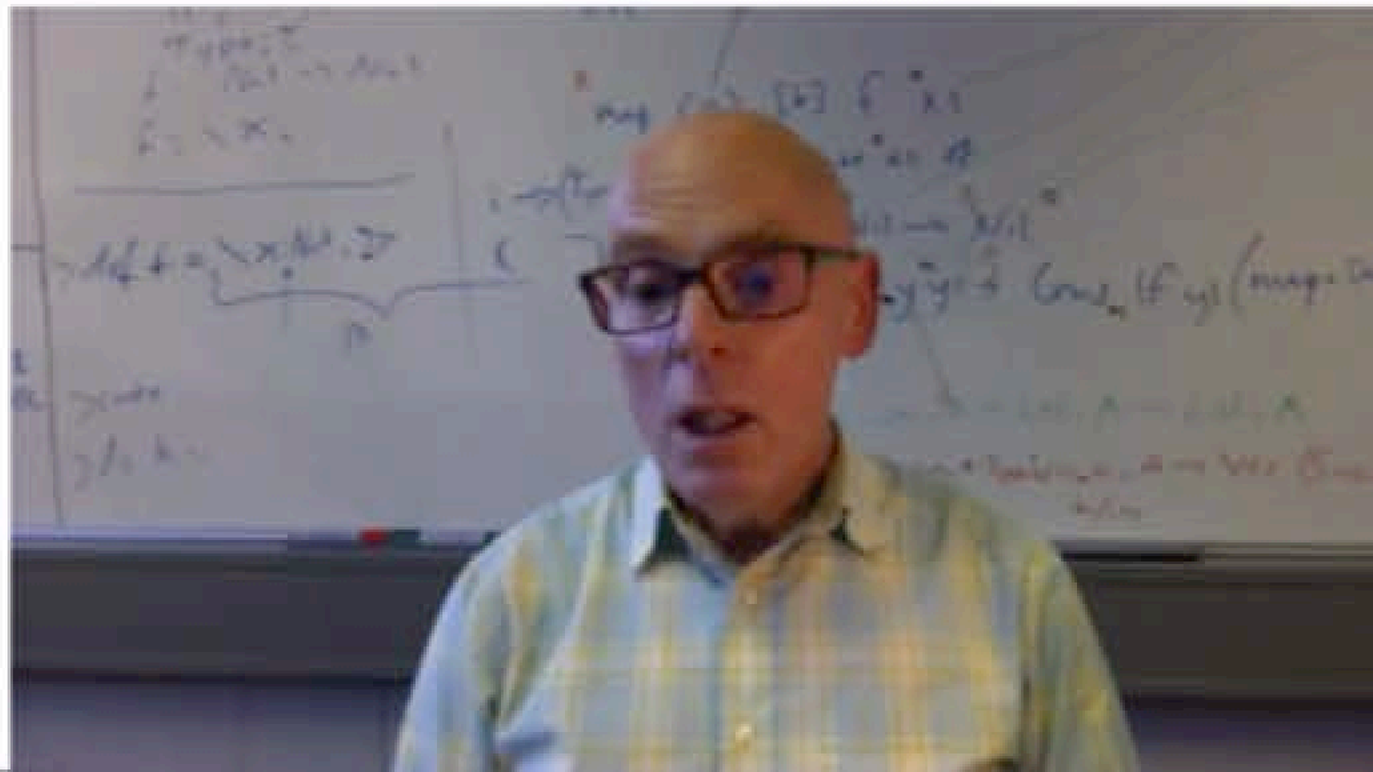
Processes can terminate normally

This process will `loop` as long as it receives messages of the form `{msg,M}` …

… and terminate if it receives anything else.

```
spawn(?MODULE,loop,[]).

loop() ->
  receive
    {msg,M} ->
      io:format("ack"),
      loop();
    _Msg ->
      ok
end.
```

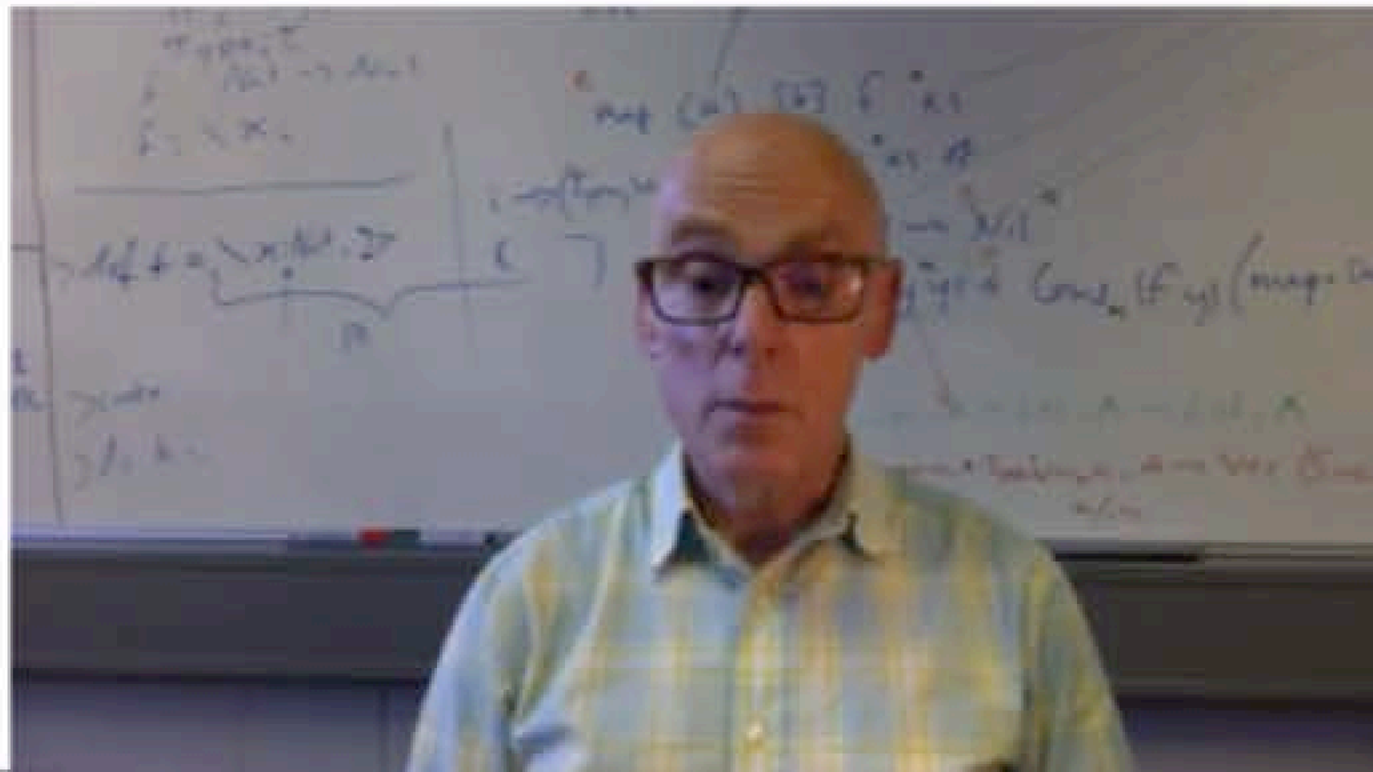


Processes can fail

This process will `loop` forever …
… and acknowledge some messages via `io`.

```
spawn(?MODULE,loop,[]).

loop() ->
  receive
    {msg,M} ->
      N=M+1,
      io:format("~b~n",[N]),
      loop();
    _Msg ->
      loop()
end.
```

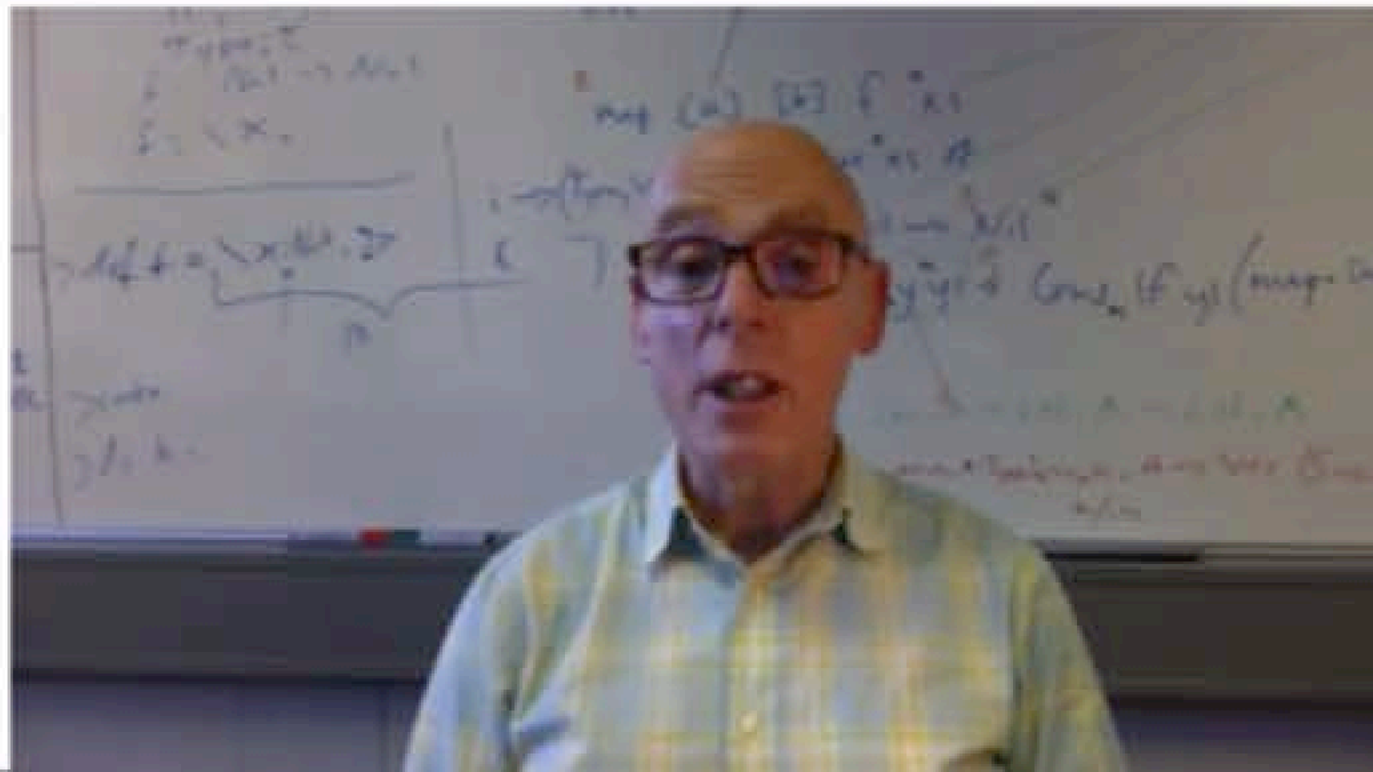


Process lifetimes

A process can execute indefinitely … and never terminate.

A process can terminate normally.

A process can fail … or terminate abnormally.



What happens when a process fails?

Take down the whole system, and – potentially – restart?

System integrity is broken …

… messages will be sent to non-existent processes …

… or awaited from a process that will never send them.

We need a mechanism to deal with this.

Respond to the failure in a more nuanced way?

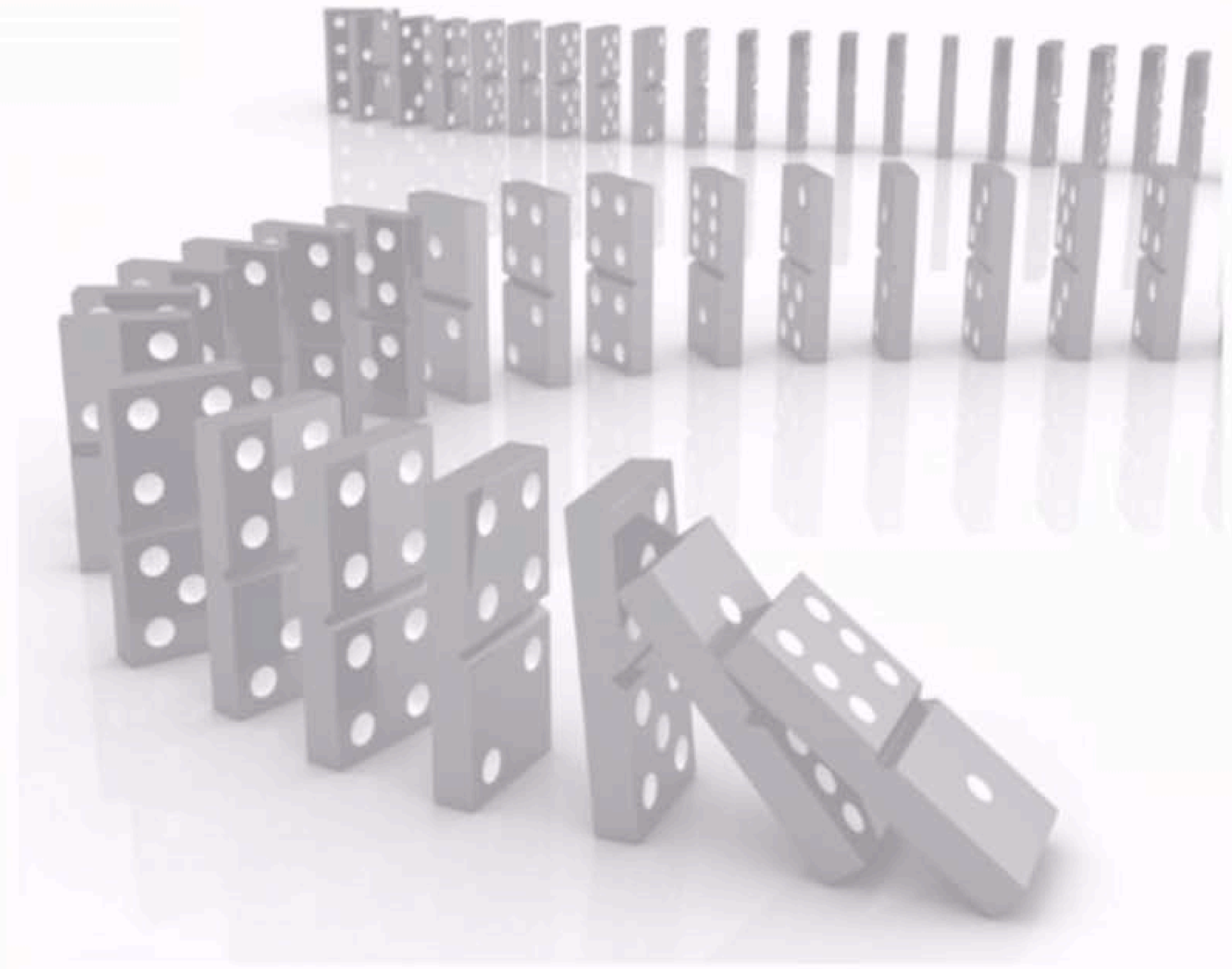
Linking processes

Call `link(Pid)` in one a process to link to …

… the process with process id `Pid`.

If one process fails, linked processes fail too …

… and processes linked to those will also fail.



http://weknowyourdreams.com/image.php?pic=/images/dominoes/dominoes-03.jpg

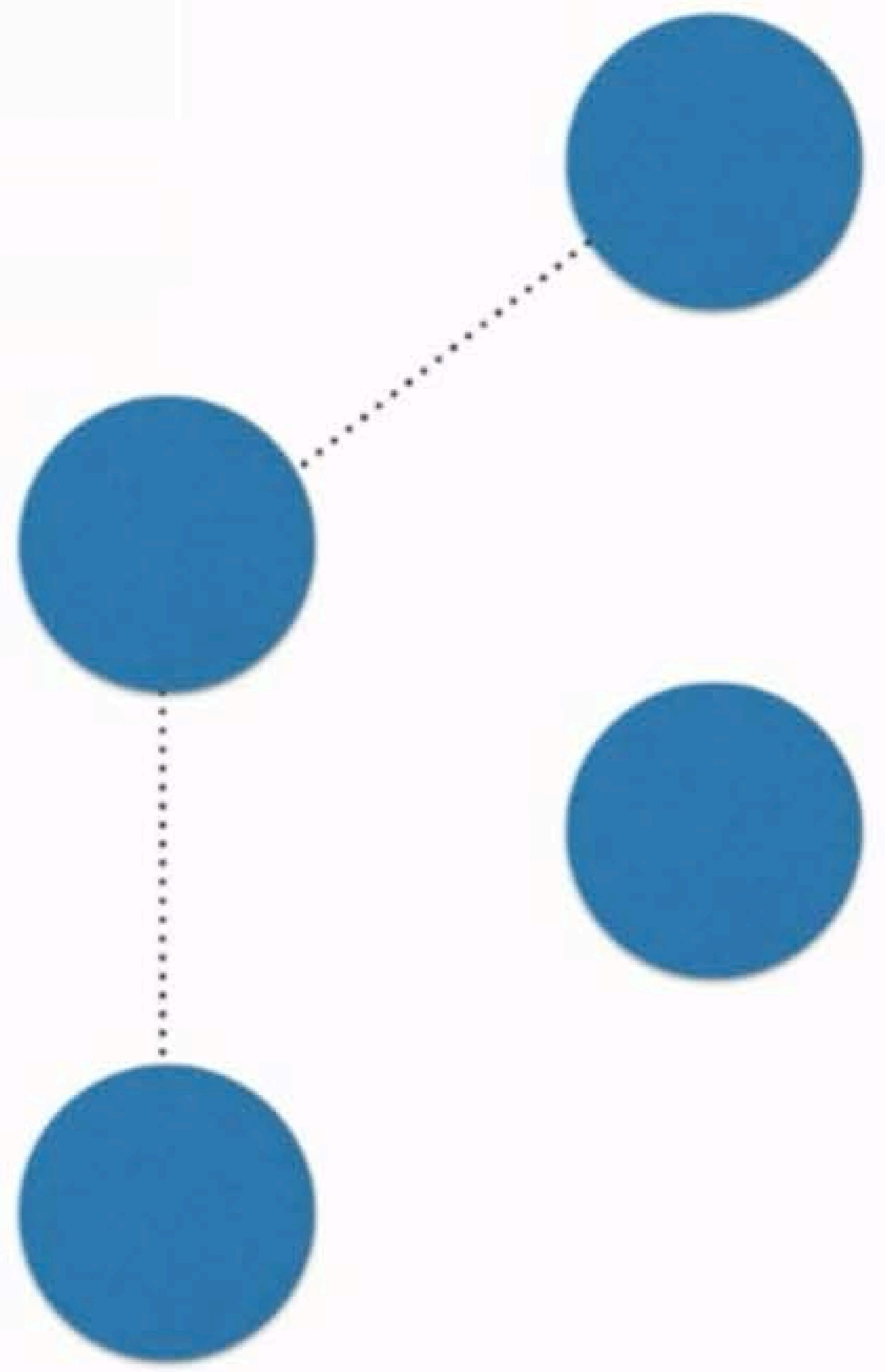
Linking processes

Call `link(Pid)` in one a process to link to …

… the process with process id `Pid`.

If one process fails, linked processes fail too …

… and processes linked to those will also fail.



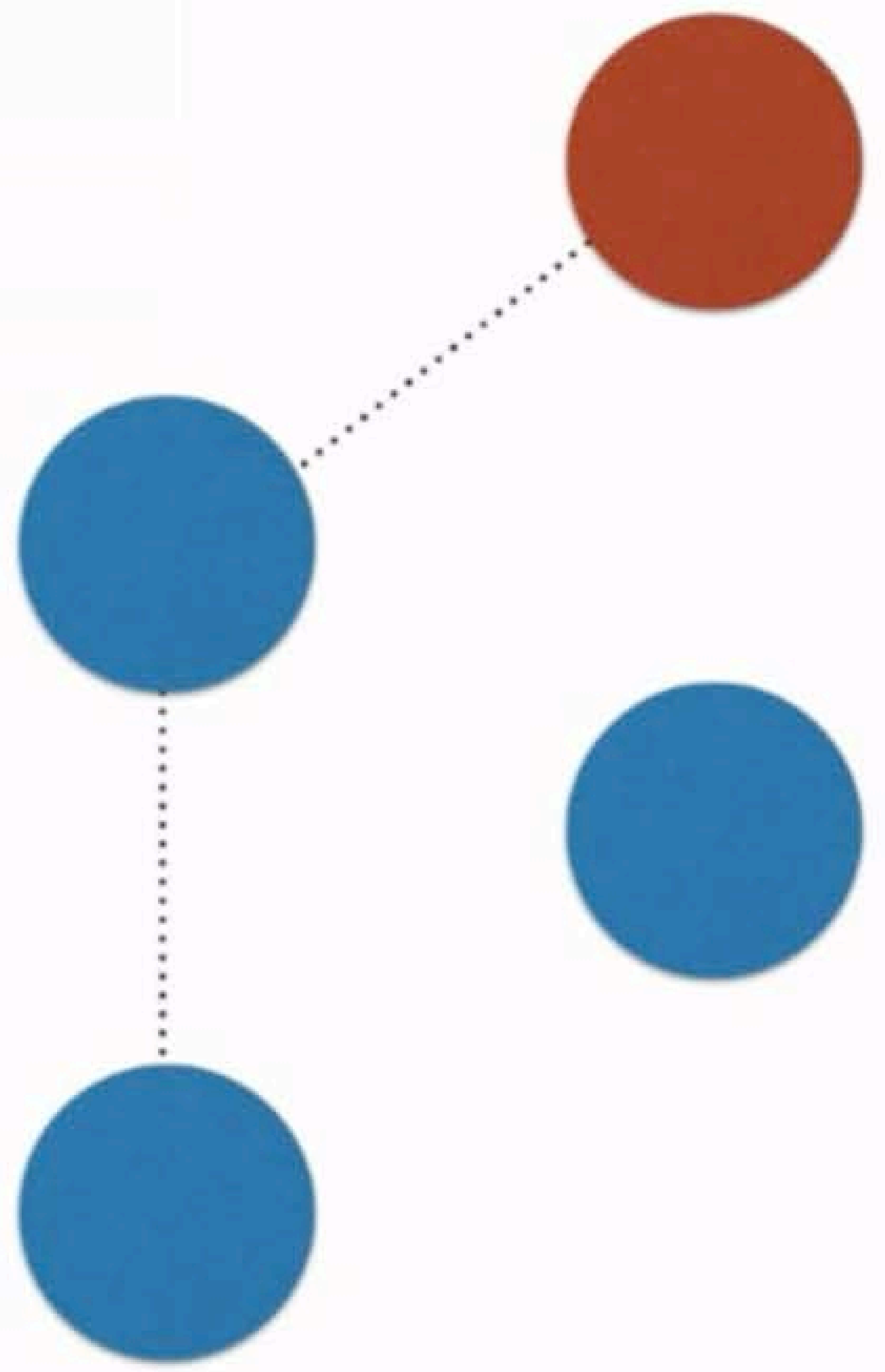
Linking processes

Call `link(Pid)` in one a process to link to …

… the process with process id `Pid`.

If one process fails, linked processes fail too …

… and processes linked to those will also fail.



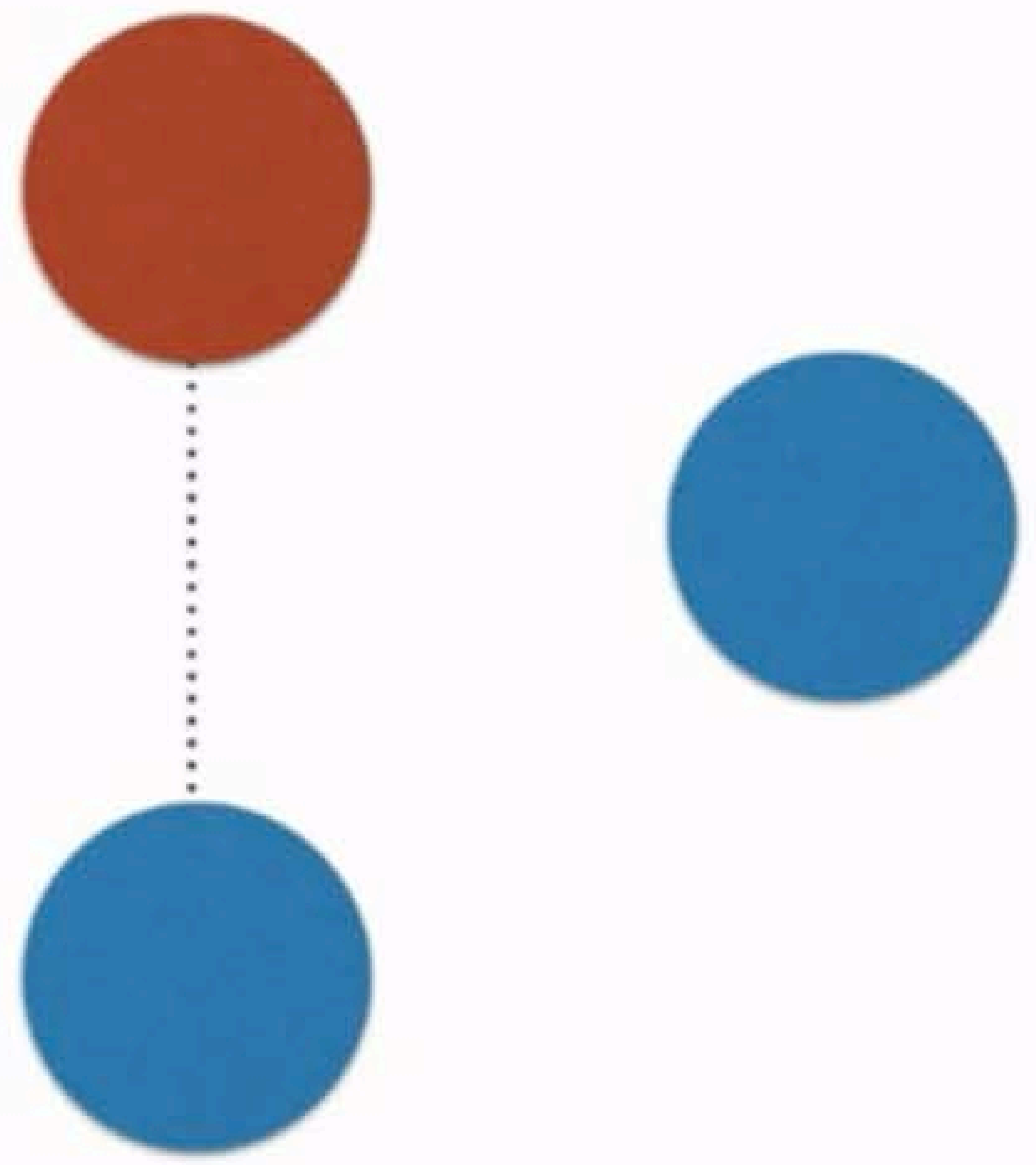
Linking processes

Call `link(Pid)` in one a process to link to …

… the process with process id `Pid`.

If one process fails, linked processes fail too …

… and processes linked to those will also fail.



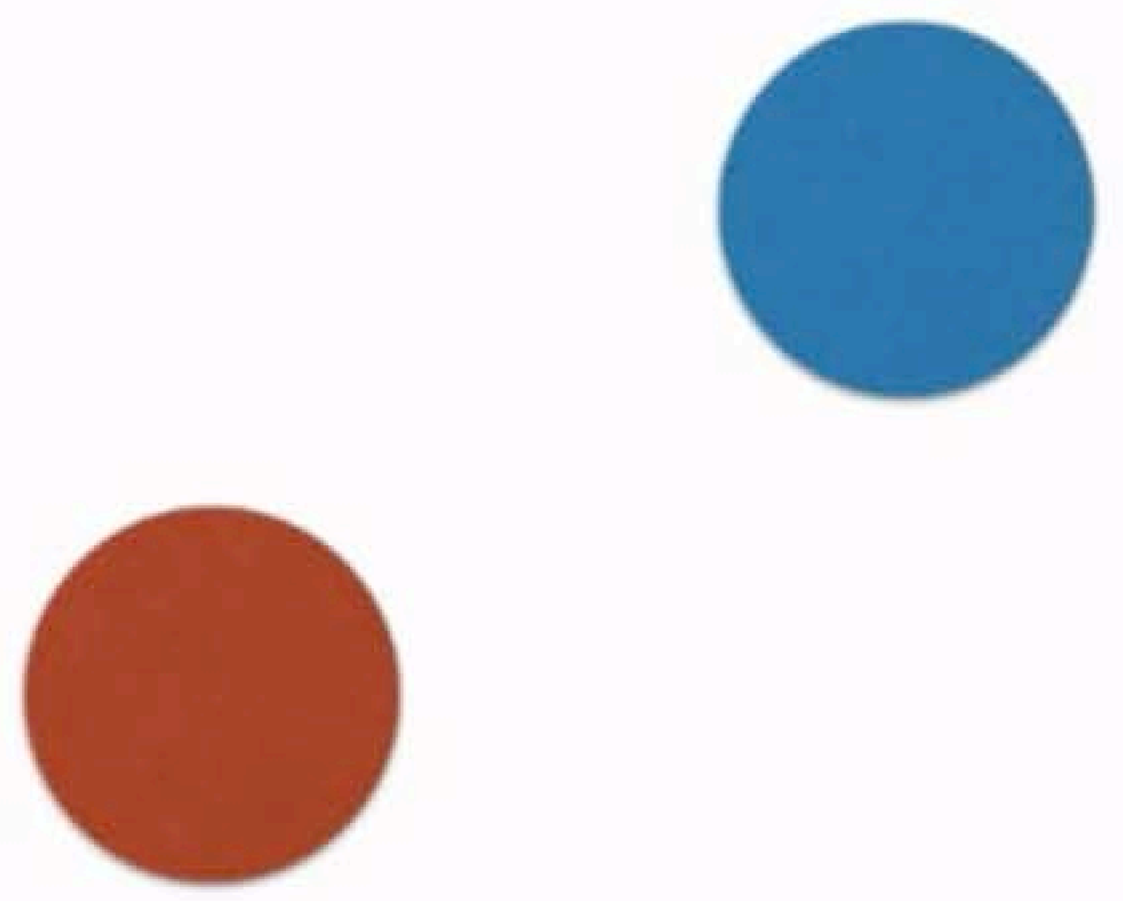
Linking processes

Call `link(Pid)` in one a process to link to …

… the process with process id `Pid`.

If one process fails, linked processes fail too …

… and processes linked to those will also fail.

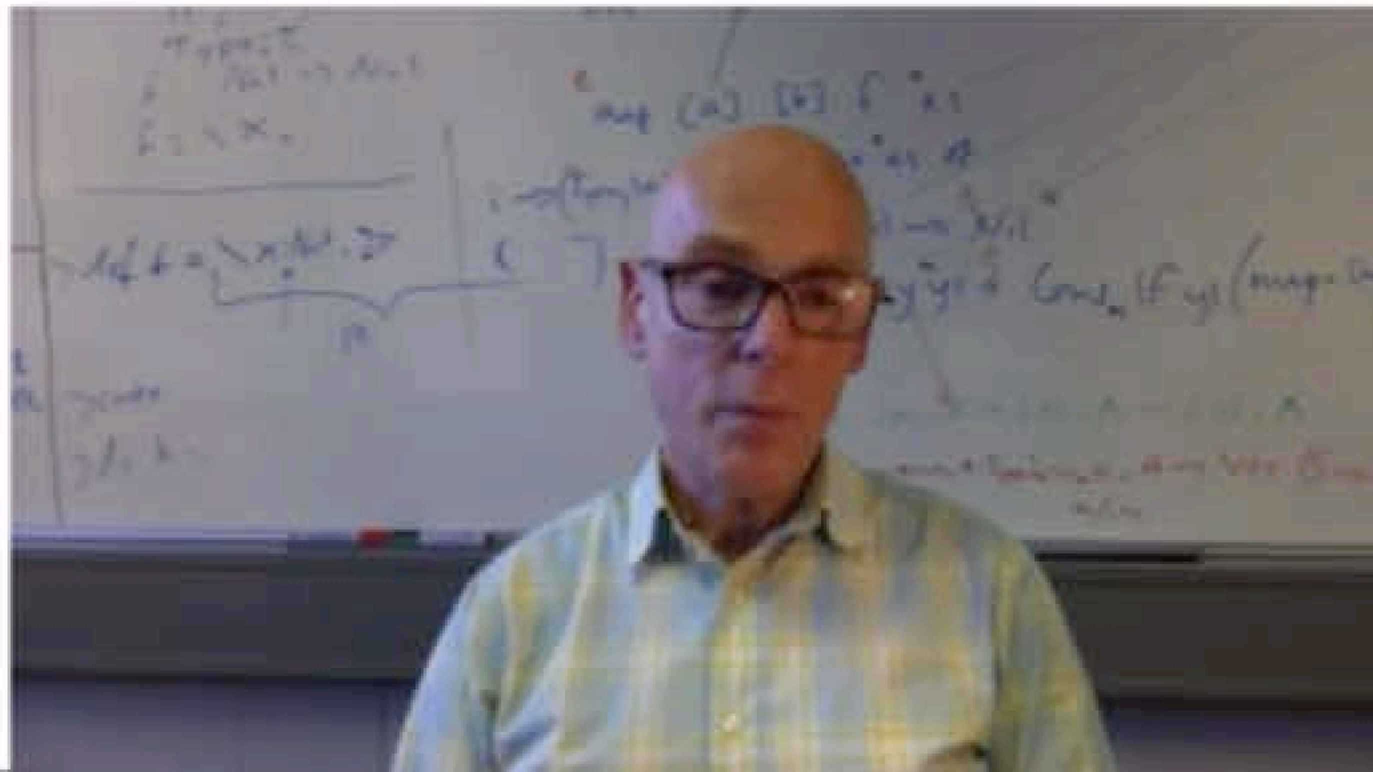


Spawning and immediately linking

We often want to link to a process that we have just spawned.

What happens if `P` dies before we're able to link to it?

```
P = spawn(?MODULE,loop,[]),
link(P),
 …
```



Spawning and immediately linking

We often want to link to a process that we have just spawned.

```
P = spawn(?MODULE,loop,[]),
link(P),
 …
```

What happens if `P` dies before we're able to link to it?

Better to `spawn_link`, which acts atomically.

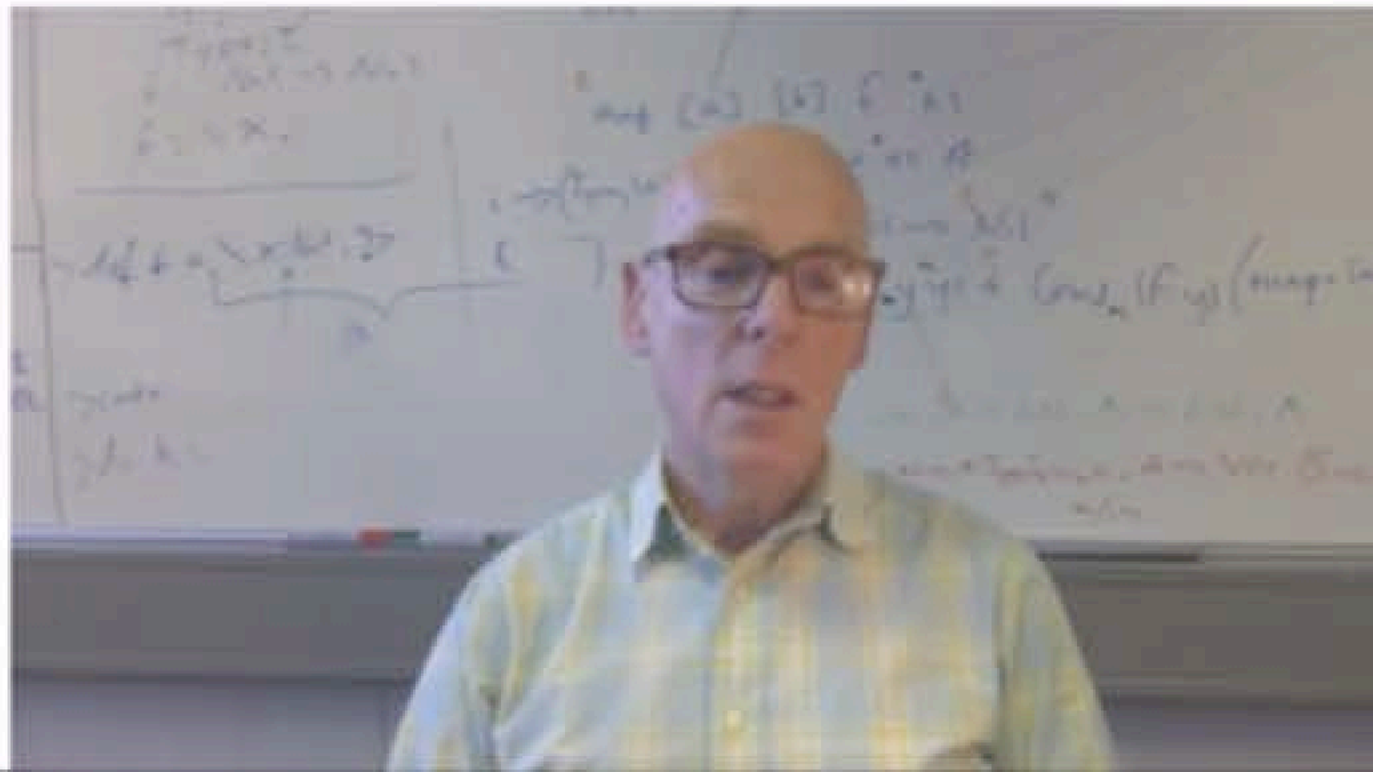
```
P = spawn_link(?MODULE,loop,[]),
 …
```

Recovering from failure

Using the linking mechanism allows us to clean up a system that has partially failed … by taking the whole thing down.

In the next step we'll hear Joe Armstrong's take on failure …

… and after that see how linking works in more detail, and show how we can "trap" exit signals and exert more control.



University of Kent