Lecture 10

Note Title 2/12/2008

Programming Erlang - Joe Armstrong

Making Reliable Distributed Systems in the presence of Software errors.

Cost of light weight processes : 2.4 GHz Celeron
512 MB

10 µs thread creation
supports > 200,000 threads w/o swapping
< 1K bytes / thread actually < 2K bytes

Mech. for high availability —

Redundancy — quick recovery from errors — F.T.
Fewer errors in code

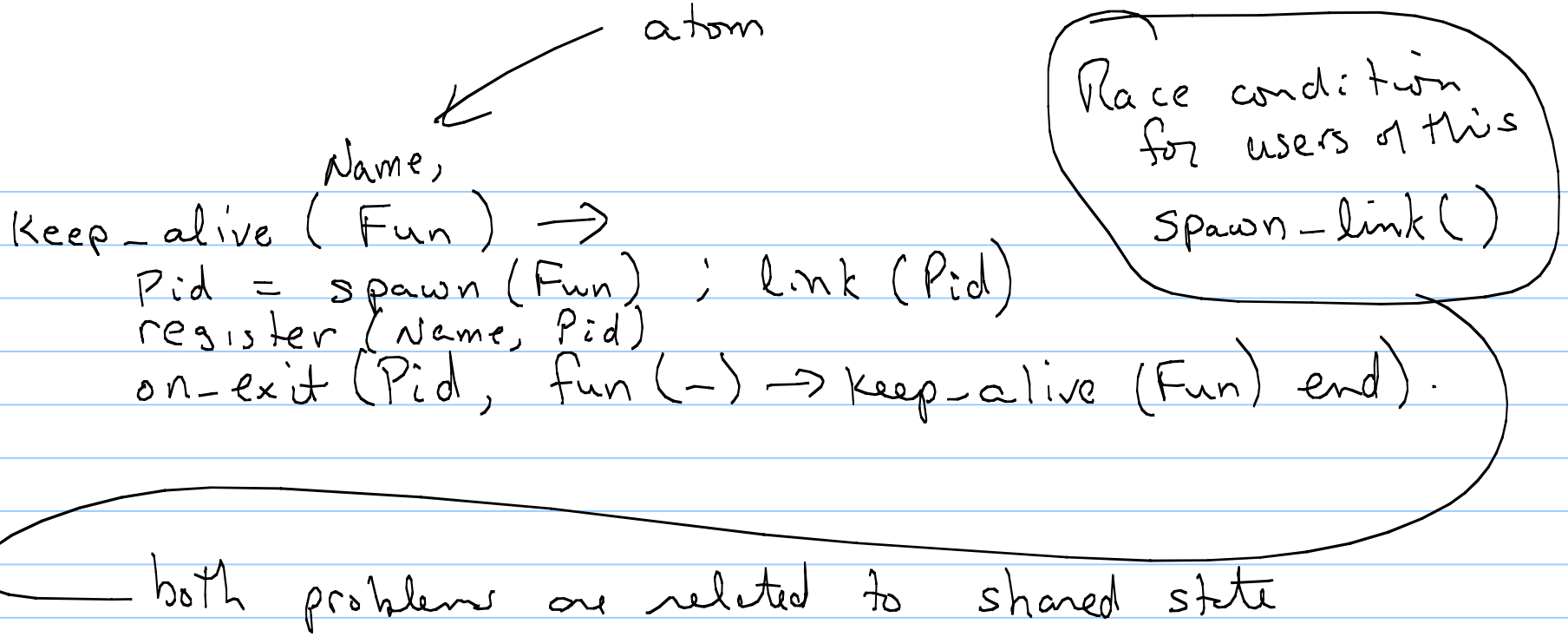
a process can link itself to another processes.

Processes that are linked form an equivalny class.
when 1 dies, all die

- Some processes are "system processes" —

they receive message in their mailbox

```
link(Pid)
receive
    {'EXIT', Pid, Why} => do something
end.
```



both problems are related to shared state

Client can ask about registration

Whereis (atom) → Pid | undefined

exit (Why)

exit (Pid, Why)

OTP

idea: behavior providing

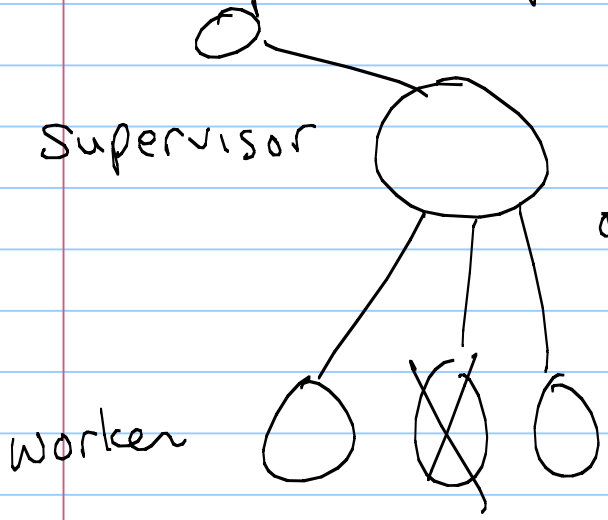
- fault tolerance
- scalability
- dynamic upgrade

} non-functional part of a system

functionality is provided by a call back module

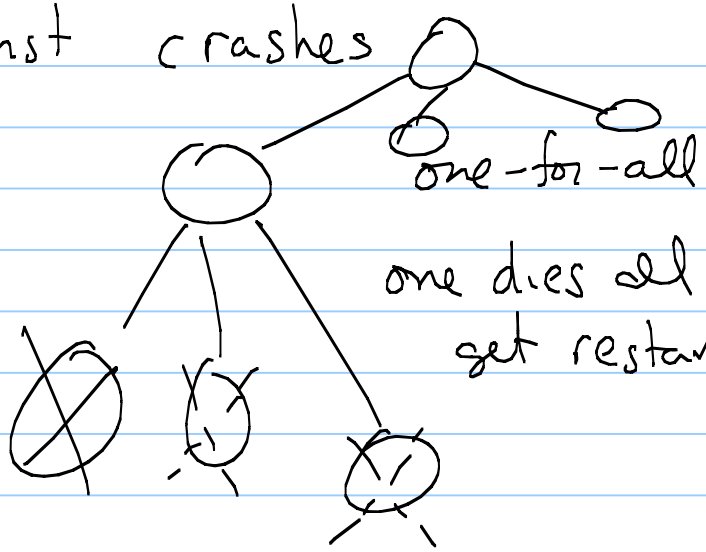
passed as a parameter to the behavior

Supervision - protection against crashes



one-for-one

one fails it gets restarted



one-for-all

one dies all get restarted

gen_supervisor (generic supervisor)

a call back module provides details about what to supervise and how.

in call back : Strategy

init() → {ok, {one-for-one | one-for-all, MaxRestarts, Time}

[Workers]

}

Workers – (Mod, Func, Args) and info about criteria for restarting

Erlang & multicore CPUs

Challenge: keep the cores busy

Advice

- Use lots of processes
- Avoid side-effects (shared state)
- Avoid sequential bottlenecks
- Use small messages to trigger big computations