CptS 483/580 March 7, 2011

Shared State:

Server process; send it messages to retrieve and change the state.

in a functional language we use a particular programming technique - tail recursion - and parameter passing to implement the state.

```
init () -> server ([])

server (State) ->
    receive
        {Pid, {get}} -> Pid ! State, server (State)
      ; {Pid, {set, NewState}} -> Pid ! State, server (NewState)
      ; {Pid, {update, UpdateFun}} -> Pid ! State,
                                      server (UpdateFun (State))
    end.

register (myInt, spawn (fun init/0)),
myInt ! {self(), set, 0},        bug here
myInt ! {self(), update, fun (X) -> X+3 end}    bug here.

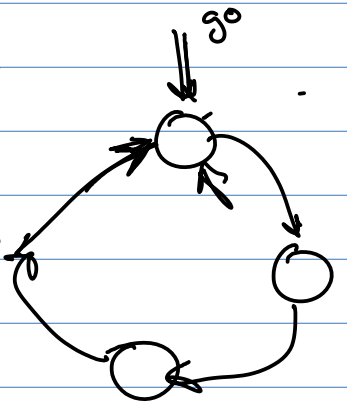
rpc (ServerName, Msg) ->
        ServerName ! {self(), Msg}
        receive
            Reply -> Reply
        end.
```

local function of the module.

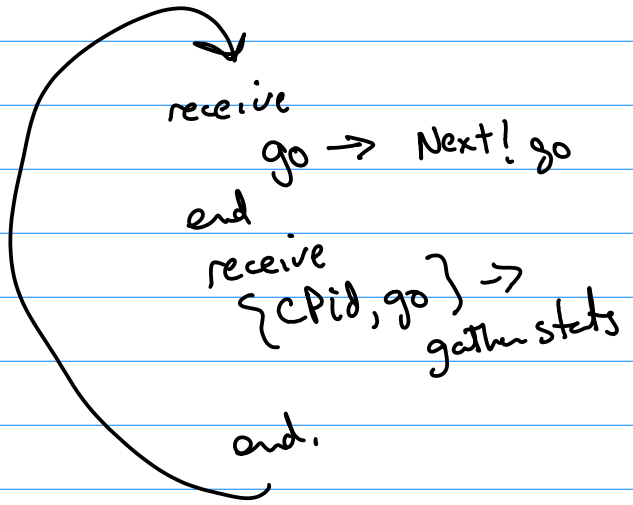
```
get (ServerName) -> rpc (ServerName, {get}),
set (ServerName, NS) -> rpc (ServerName, {set, NS})
update (ServerName, UpdateFun) -> rpc (ServerName, {update, UpdateFun})
```

```
rpcWithTimeout (ServerName, Msg, Timeout) ->
        Tag = make_ref(),
        ServerName ! {self(), Msg, Tag},
        receive
            {Tag, Reply} -> Reply
            after Timeout -> Server ! {Tag, cance}, timedOut
        end.
```

Ring assignment




```
receive ->
  {cPid, go} -> gatherStats
; {pPid, go} -> Next! go
end.
```



```
receive
    go -> Next! go
end
receive
  {cPid, go} ->
      gatherStats
end.
```