Industrial Functional Programming [1]

Melinda Tóth, István Bozó

Dept. Programming Languages and Compilers
Eötvös Loránd University, Budapest, Hungary

[1] Supported by TÁMOP-4.1.2.A/1-11/1-2011-0052

Contents

1 Process Registration

2 Disributed Erlang

Registering Processes

- Giving names/aliases to processes
- Names are atoms
- Processes can be referred by the atoms
- Registration:

```
register(Alias, Pid)
register(foo, pid(0,30,0))
```

Message Sending

- Message sending:
  `foo ! {msg, "Final message”}`
- Release the name: `unregister(Alias)`
- VM commands: `registered(), regs()`
- BIF: `whereis(Alias)`

Distributed Erlang Nodes

- Node names are atoms
- Unique names required
- Starting the node:

```
erl -name foo@host
erl -sname foo -setcookie bar
```

- BIF: `node()`
- ps ax | grep -i epmd
- Erlang Port Mapper Daemon

Distributed Erlang Nodes

- Connecting the nodes:

  ```
  net_adm:ping(Node),
  monitor_node(Node, true)
  erlang:monitor_node(Node)
  ```
- BIF: `nodes()`
- "Magic Cookie":

  ```
  get_cookie(), set_cookie(node(), "bar")
  ```
- Remote shell: `Ctrl-G: r, c`
- Default transitive connection

Distributed Erlang Nodes

- Default transitive connection
- For non transitive connection use the flag `connect_all false`
- Hidden nodes: when `-hidden` flag used
- The hidden connections are not transitive
- BIF: `nodes(hidden), nodes(connected)`

Distributed Erlang Nodes

- Message passing:
  ```
  {Name, Node} !  {msg, "Final message”}
  Pid !  {msg, "Final message”}
  ```
- Spawning:
  ```
  spawn(Node, Mod, Fun, [Arg1, ..., ArgN]
  spawn(Node, FunExpr)
  ```
- Receiving messages: remains the same

Example Connection

Start the server node:

```
erl -name server@188.143.112.97
```

Start the client nodes:

```
erl -name client1@188.143.112.97
erl -name client2@188.143.112.97
```

Example Connection

Connect to the nodes:

```
(server@188.143.112.97)5>
    net_adm:ping('client1@188.143.112.97').
pong
(server@188.143.112.97)6> nodes().
['client1@188.143.112.97']
(server@188.143.112.97)5>
    net_adm:ping('client2@188.143.112.97').
pong
(server@188.143.112.97)6> nodes().
['client1@188.143.112.97','client2@188.143.112.97']
```

Distributed Ping-Pong

```
run() ->
  Pid = spawn(mynode@localhost, fun ping/0),
  Pid ! {ping, self()},
  receive
    pong -> ok
  after
    1000 -> nok
  end.

ping() ->
  receive
    {ping, From} -> From ! pong
  end.
```

Tools to Use

- `pman:start()`
- `tv:start()`
- `appmon:start()`

On the Next Lecture ...

- Server Skeleton
- Software Upgrade
- ETS/DETS