Control Systems Optimization

Igor Wojnicki

AGH – Univeristy of Science and Technology

2010

Outline

Open Telecom Platform: OTP

- Behavior.
- Encapsulating common behavioral patterns:
  - General Server
  - Finite State Machine
  - Event Handler
  - Supervising Tree
  - Application
- Fault Tolerance, Scalability, Dynamic Code Upgrade.
- Solves non functional parts of the problem.
- Structure: Behavior + Callback.

gen_server, Intro

- Decide on a callback module name.
- Write the interface functions.
- Write six required callback functions in the callback module.

gen_server, Steps 1,2,3

- Module: `iwd`, with the following functions:
  - add(Key,Value)
  - del(Key)
  - lookup(Key)
  - stop() – optional
- Callbacks:
  - `init/1`, `handle_call/3`, `handle_cast/2`, `handle_info/2`, `terminate/2`, `=code_change/3`

supervisor

- Supervisors and Workers
- Monitor children
- Take action when they terminate
  `{ok, {SupervisorSpec, ChildrenSpec}}`

Supervisor Specs

```
{RestartStrategy, AllowedRestarts, MaxSeconds}
```

- RestartStrategy:
  - `one_for_one` – restart the misbehaving child,
  - `one_for_all` – terminate all children and restart,
  - `rest_for_one` – terminate all children started after the one that crashed and restart

Child Specs

```
{Id, {Module, Function, Arguments}, Restart, Shutdown, Type,
ModuleList}
```

- Id – unique child's id within the supervisor
- Restart – transient (never restart), temporary (restart on crash), permanent (always restart)
- Shutdown – timeout for terminate() within behavior
- Type – worker or supervisor
- ModuleList – modules implementing the process

Supervisor: Dynamic Children

```
supervisor:start_child(SupervisorName, ChildSpec)
supervisor:terminate_child(SupervisorName, Id)
supervisor:restart_child(SupervisorName, Id)
supervisor:delete_child(SupervisorName, Id)
```

- SupervisorName – supervisor's Pid or name
- ChildSpecs – as before
- Id – unique child's id

Supervisor: simple one to one

```
-module(simple_sup). -behaviour(supervisor).
-export([start_link/0]). -export([init/1]).
start_link() ->
    supervisor:start_link(simple_sup, []).
init(_Args) ->
    {ok, {{simple_one_for_one, 0, 1},
          [{call, {call, start_link, []},
            temporary, brutal_kill, worker, [call]}]}}.
```

- No children are started upon supervisor's start.
- All child processes are added dynamically by calling:
  `supervisor:start_child(Sup, List)`
  - Sup is the pid, or name, of the supervisor.
  - List is a list of terms which will be added to the list of arguments specified in the child specification.

application

OTP Application – a component that could be started and stopped as a unit, regardless of the number of processes it consists of.

- Application description: `*.app` file
  `{application, Application, [Opt1,...,OptN]}.`
- Callback module.
  - start()
  - stop()

Running Applications

- `application:loaded_applications()`
- `application:load(appname)`
- `application:start(appname)`
- `application:stop(appname)`

gen_fsm

Finite State Machine

gen_event

Event Handler