CS 683 Emerging Technologies
Fall Semester, 2008
Doc 8 Erlang Applications
Sept 25 2008

References

Programming Erlang: Software for a Concurrent World, Armstrong, Chapter 18

Appmon Users Guide, http://www.erlang.org/doc/apps/appmon/part_frame.html

SASL Reference Manual, http://www.erlang.org/doc/apps/sasl/index.html

EDoc User's Guide, http://www.erlang.org/doc/apps/edoc/part_frame.html

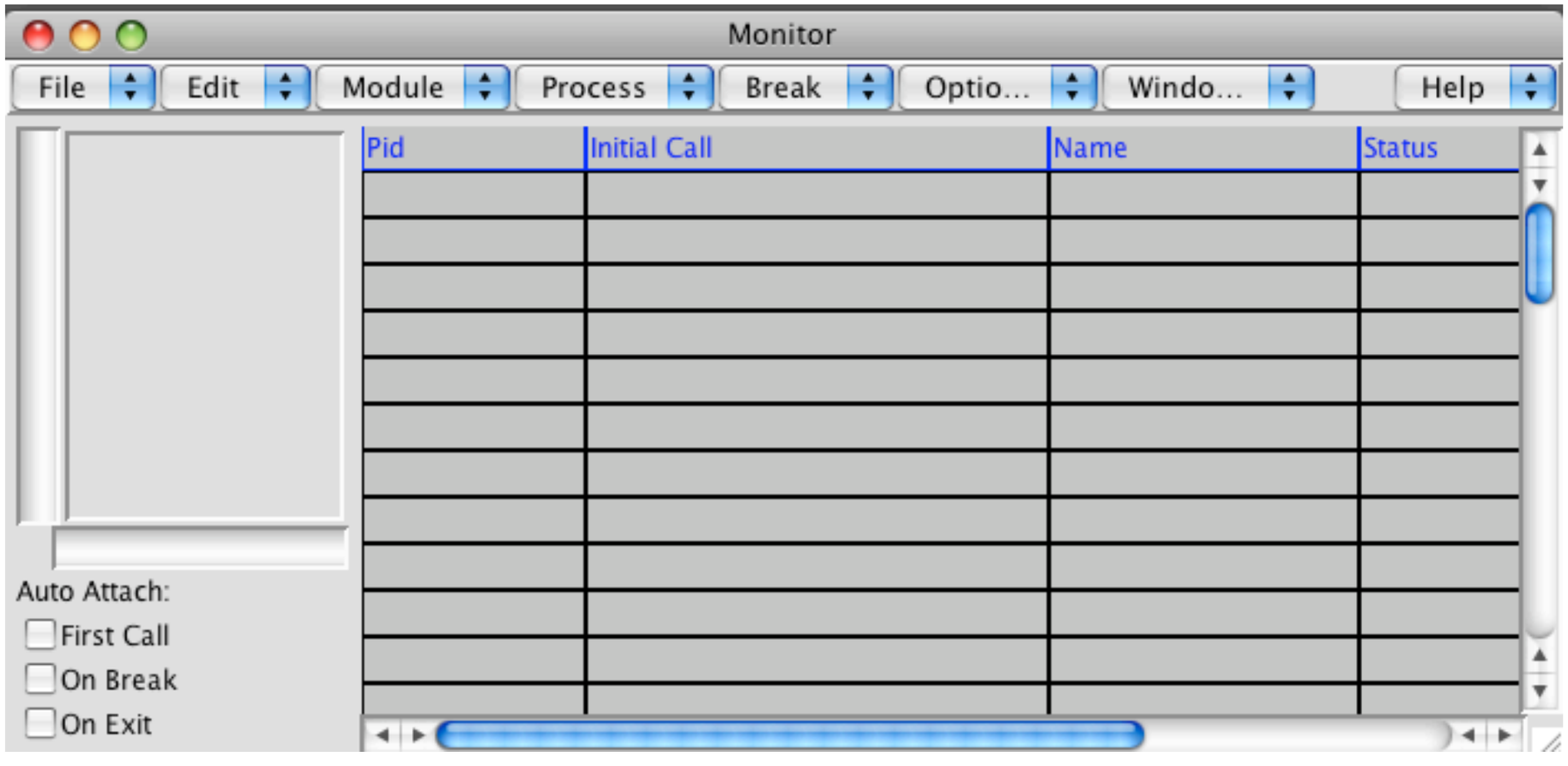
Using Erlang Debugger

Step one Compile with the debug_info option

```
c(my_module, [debug_info]).
```

Step two Run the debuger

```
debugger:start().
```

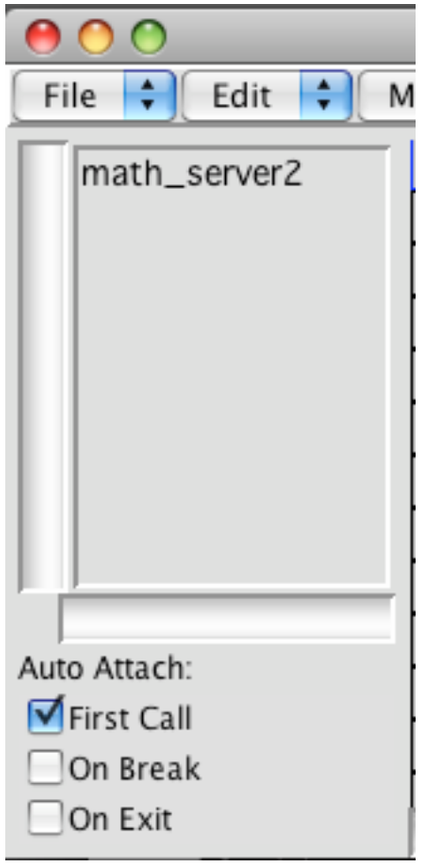


Using Erlang Debugger

Step 3 Use Module menu to load your module

Step 4 Select First Call

Step 5 Call a function



EDoc

```
%% @author Roger Whitney <whitney@cs.sdsu.edu>
%% @copyright 2008 Roger Whitney
%% @doc This module contains some really <em>bad</em> code. It is used
%% in a demo of the Erlang Debugger.

-module (bad_math).
-export([ factorial/1]).

%% @spec factorial (1) -> 1
factorial(1) ->
      Answer = 1 + 1,
      Answer - 1;

%% @throws factorialNonPositiveArgument
factorial(N) when N < 1 ->
      throw({factorialNonPositiveArgument, N});

%% @spec factorial (N::integer()) -> integer()
factorial(N) ->
```

EDoc

Running

```
2> edoc:files(["bad_math.erl"]).
```

See

http://www.erlang.org/doc/apps/edoc/part_frame.html

Production Runtime

```
Al pro 47->erl -boot start_sasl
Erlang (BEAM) emulator version 5.6.3 [source] [smp:2] [async-threads:0] [kernel-poll:false]

=PROGRESS REPORT==== 25-Sep-2008::12:47:11 ===
        supervisor: {local,sasl_safe_sup}
           started: [{pid,<0.34.0>},
                     {name,alarm_handler},
                     {mfa,{alarm_handler,start_link,[]}},
                     {restart_type,permanent},
                     {shutdown,2000},
                     {child_type,worker}]
etc.
=PROGRESS REPORT==== 25-Sep-2008::12:47:11 ===
        application: sasl
         started_at: nonode@nohost
Running ErlangEshell V5.6.3  (abort with ^G)
1>
```

SASL

System Architecture Support Libraries

error logging
alarm handling
overload regulation
release handling
report browsing

http://www.erlang.org/doc/apps/sasl/index.html

SASL Error Logging

Supervisor Report

A supervised child terminates in an unexpected way

Progress Report

A supervisor starts or restarts

Crash Report

A process terminates with an unexpected reason

Logging

```
2> error_logger:error_msg("Bad News\n").
ok

=ERROR REPORT==== 25-Sep-2008::13:08:01 ===
Bad News

3> error_logger:logfile({open, "sampleLog"}).
ok
4> error_logger:warning_msg("Bad News\n").
ok

=ERROR REPORT==== 25-Sep-2008::13:11:36 ===
Bad News

5> error_logger:tty(false).
ok
6> error_logger:warning_msg("Bad News\n").
ok
7>
```

```
error_logger:error_msg("Bad News\n")
error_logger:warning_msg("Bad News\n")
error_logger:info_msg("Bad News\n")
error_logger:logfile({open, "sampleLog"})
error_logger:tty(false)
```

Log Configuration File

One can create a log configuration file

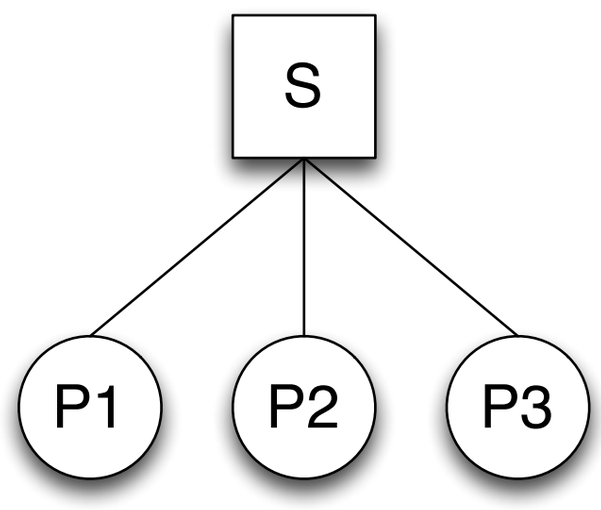
You specify the configuration file when you start erl

Supervision Trees

Supervisors restart processes that fail

One-for-one Supervision Tree

Supervisor restarts the process that fails



All-for-one Supervision Tree

If one process fails supervisor
- kills all the rest
- Restarts all process

math_server.erl

```
-module (math_server).
-behaviour (gen_server).
-export([start/0, stop/0, factorial/1]).
-export([init/1, handle_call/3, handle_cast/2, handle_info/2,
        terminate/2, code_change/3]).

start() -> gen_server:start_link({local, ?MODULE}, ?MODULE, [], []).
stop()  -> gen_server:call(?MODULE, stop).

factorial(N)      -> gen_server:call(?MODULE, {factorial, N}).

init([]) ->
      io:format("~p starting~n", [?MODULE]),
      {ok,1}.

handle_call({factorial,N}, _From, Count) ->
      {reply, {factorial,compute_factorial(N),count, Count}, Count + 1};
```

math_server.erl

```
handle_call(stop,_From, Count) ->{stop, normal, server_stopped, Count}.

compute_factorial(1) -> 1;
compute_factorial(N) when N < 1 ->
    throw({factorialNonPositiveArgument, N});
compute_factorial(N) ->
    N * compute_factorial(N - 1).


handle_cast(_Msg, State) -> {noreply, State}.
handle_info(_Info, State) ->    {noreply, State}.
terminate(_Reason, _State) ->
    io:format("terminating ~p~n", [?MODULE]),
    ok.
code_change(_OldVsn, State, _Extra) -> {ok, State}.
```

math_supervisor.erl

```
-module(math_supervisor).
-behaviour(supervisor).

-export([start/0, start_in_shell_for_testing/0, start_link/1, init/1]).

start() ->
    spawn(fun() ->
            supervisor:start_link({local,?MODULE}, ?MODULE, _Arg = []) 
        end).

start_in_shell_for_testing() ->
    {ok, Pid} = supervisor:start_link({local,?MODULE}, ?MODULE, _Arg = []),
    unlink(Pid).

start_link(Args) ->
    supervisor:start_link({local,?MODULE}, ?MODULE, Args).
```

math_supervisor.erl

```
init([]) ->
    {ok, {{one_for_one, 3, 10},
        [{math_super,
          {math_server, start, []},
          permanent,
          10000,
          worker,
          [math_server]}
        ]}}.
```

init explained

```
init([]) ->
   {ok, {{RestartStrategy, MaxR, MaxT},},
...
```

RestartStrategy

one_for_one

one_for_all

If MaxR restarts occur in MaxT seconds supervisor and child processes are killed

init explained

```
[{math_super,
        {math_server, start, []},
        permanent,
        10000,
        worker,
        [math_server]}]
```

{Id, StartFunc, Restart, Shutdown, Type, Modules}

Id - id of worker

StartFunc - function to start worker

Restart

- permanent - always restart
- temporary - never restart
- transient - restart if abnormally ends

Shutdown

- brutal_kill - use exit(Child, kill) to terminate workers
- integer - use exit(Child, shutdown)

Type

- worker
- supervisor

Modules - ones supervisor uses

Running the Example

```
2> math_supervisor:start_in_shell_for_testing().
math_server starting
true
3> math_server:factorial(4).
{factorial,24,count,1}
4> math_server:factorial(-4).
terminating math_server
=ERROR REPORT==== 25-Sep-2008::13:30:53 ===
** Generic server math_server terminating
** Last message in was {factorial,-4}
** When Server state == 2
** Reason for termination ==
** {bad_return_value,{factorialNonPositiveArgument,-4}}
math_server starting
** exception exit: {{bad_return_value,{factorialNonPositiveArgument,-4}},
                    {gen_server,call,[math_server,{factorial,-4}]}}
     in function  gen_server:call/2
5> math_server:factorial(4).
{factorial,24,count,1}
```

Applications

Component implementing some specific functionality
Can be started and stopped as a unit
Can be re-used in other systems
Generate EDoc per application

mathstore.app

```
{application, mathstore,
 [{description, "The Math Store"},
  {vsn, "1.0"},
  {modules, [mathstore_app, math_supervisor, math_server]},
  {registered,[math_server, math_supervisor]},
  {applications, [kernel,stdlib]},
  {mod, {mathstore_app,[]}},
  {start_phases, []}
 ]}.
```

mathstore.app Explained

vsn - Version number

modules

All modules introduced by this application

A module must be defined in one and only one application

registered

Names of registered processes.

applications

All applications which must be started before this application

```
{application, mathstore,
 [{description, "The Math Store"},
  {vsn, "1.0"},
  {modules, [...]},
  {registered,[...]},
  {applications, [kernel,stdlib]},
  {mod, {mathstore_app,[]}},
  {start_phases, []}
 ]}.
```

mathstore_app.erl

```
-module(mathstore_app).
-behaviour(application).
-export([start/2, stop/1]).

start(_Type, StartArgs) ->
    math_supervisor:start_link(StartArgs).

stop(_State) ->
    ok.
```

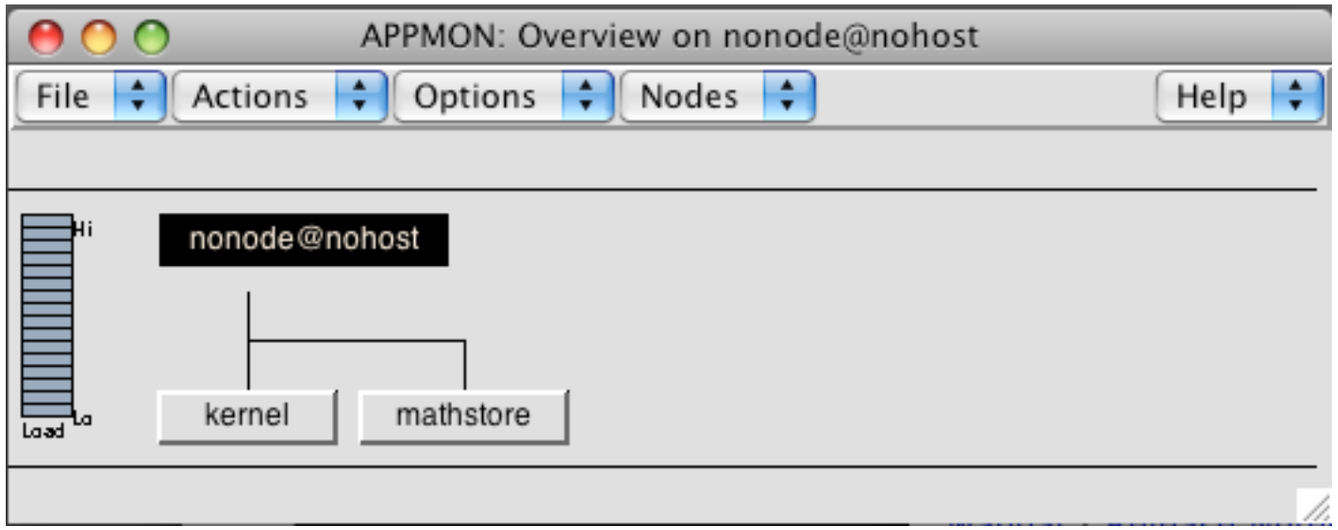
Running the Application

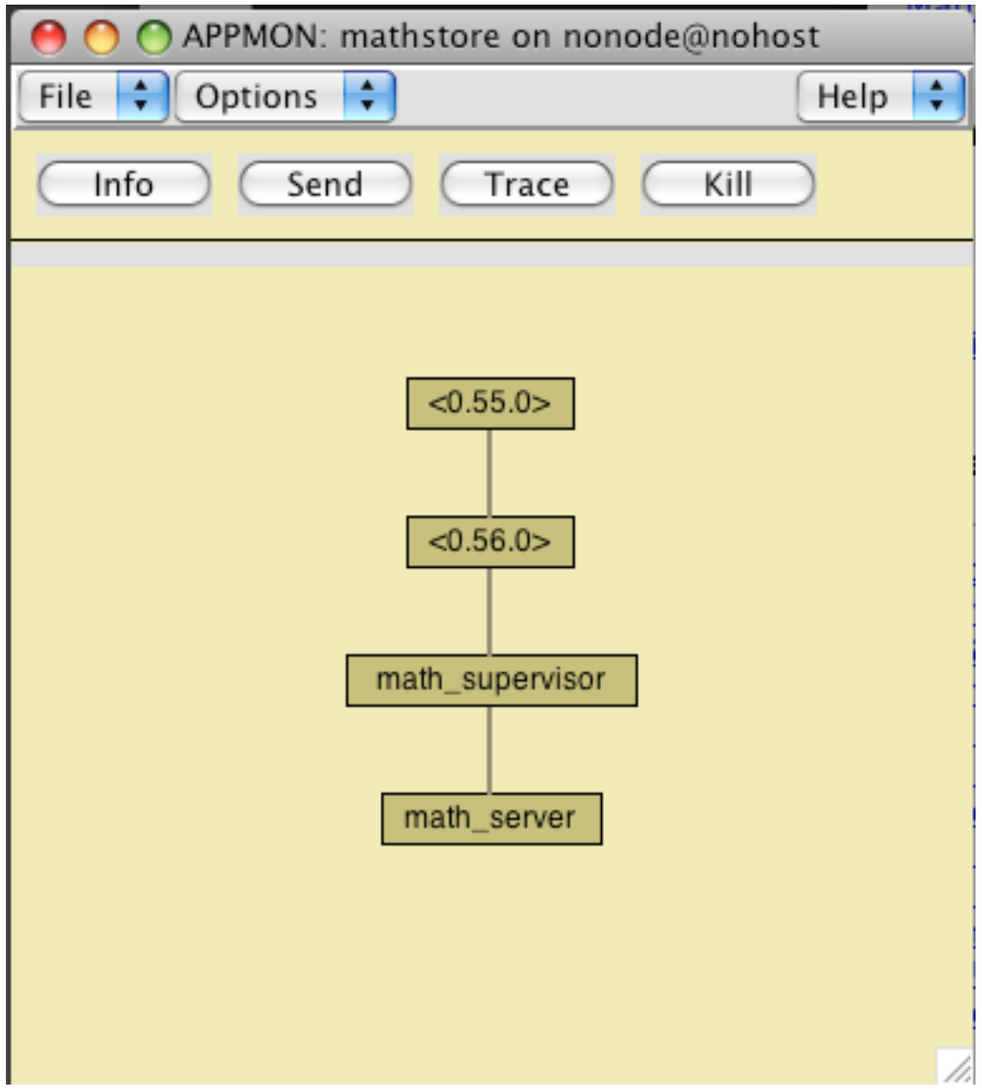
```
4> application:start(mathstore).
math_server starting
ok
5> math_server:factorial(4).
{factorial,24,count,1}
6> application:stop(mathstore).

=INFO REPORT==== 25-Sep-2008::14:27:07 ===
    application: mathstore
    exited: stopped
    type: temporary
ok
7> math_server:factorial(4).
** exception exit: {noproc,{gen_server,call,[math_server,{factorial,4}]}}
     in function  gen_server:call/2
```

App Monitor

9> appmon:start().





App Monitor Process Information

APPMON: Process Information

File

```
Node: nonode@nohost, Process: <0.58.0>
[{registered_name,math_server},
 {current_function,{gen_server,loop,6}},
 {initial_call,{proc_lib,init_p,5}},
 {status,waiting},
 {message_queue_len,0},
 {messages,[]},
 {links,[<0.57.0>]},
 {dictionary,[{'$ancestors',[math_supervisor,<0.56.0>]},
              {'$initial_call',{gen,init_it,
                                [gen_server,<0.57.0>,<0.57.0>,
                                 {local,math_server},
                                 math_server,[],[]]}}]},
 {trap_exit,false},
 {error_handler,error_handler},
 {priority,normal},
 {group_leader,<0.55.0>},
 {total_heap_size,1597},
 {heap_size,987},
 {stack_size,12},
 {reductions,473},
 {garbage_collection,[{fullsweep_after,65535},
                      {minor_gcs,28}]},
 {suspending,[]}]
```

Done