Erlang Solutions Ltd.

Modules and Processes

Modules and Processes

- Modules
- System Information
  - Processes
  - Registered Names
  - Saving Calls
- Other Process Issues

Modules

- A bug is solved and a patch is loaded in the run time system
- The bug is still there
- What went wrong?
  - Loading of the .beam file failed
  - The .beam file was not first in the code search path
  - The .beam file is not in the search path
  - The .beam file hasn't been compiled
- How do we get detailed module information?

Modules

- **m(ModuleName)**
- Provides a quick way of printing module information.
- It always picks the newest (last loaded) version.
- Information includes
  - Compile time and date
  - Location of the byte code
  - Compile options
  - Exported functions
  - If available, source file location

Modules

```
1> m(lists).
Module lists compiled: Date: June 2 2011, Time: 18.58
Compiler options:  [{no_auto_import,[{max,2}]},
                    {inline,[{merge3_12,7}, ...]
Object file: /usr/local/Cellar/erlang/R14B03/lib/erlang/
lib/stdlib-1.17.4/ebin/lists.beam
Exports:
all/2                          nthtail/2
...                            ...
```

- Was the module compiled?
- Is the module in the code search path?
- Is it first in the code search path?

Modules

- **Module:module_info()**
- Returns a list of tuples with module information.
  - Function exports
  - Module attributes
  - Compile time and date
  - Compiler version
  - Compiler options such as
    - macro directives
    - Include paths
    - Working directory during compilation
    - Out directory for compiled modules

Modules

```
%%% Purpose : The resource server
-module(server).
-author('trainers@erlang-solutions.com').
-behaviour(gen_server).
-vsn(r7a).

%% Exports
-export([start_link/0, stop/0, allocate/0, deallocate/1]).
-export([init/1, handle_call/3, terminate/2]).

%%% API
start_link() ->
    gen_server:start_link({local,?MODULE}, ?MODULE,[],[]).
```

Module attributes

Exported functions

Modules

```
1> server:module_info().
[{exports,[{start_link,0}, {stop,0}, {allocate,0},
           {deallocate, 1}, {init,1},{handle_call,3},
           {terminate,2}]},
 {imports,[]},
 {attributes,[{author,['trainers@erlang-solutions.com']},
              {behaviour,[gen_server]},
              {vsn,[r7a]}]},
 {compile,[{options,[{cwd,"/Users/.../"},
                     {outdir,"/Users/.../"}]},
           {version,"4.7.4"},
           {time,{2011,9,7,12,34,30}},
           {source,"/Users/.../server.erl"}]}]
```

Exported functions

Module attributes

Compile time, options and versions

System Information

- A system is running in embedded mode.
  - It can only be accessed through an ssh session.
- You get intermittent crashes at large (days) intervals.
- The logs are too large and generic to be of any use.
- The existing debugging tools provide no help.
- How to retrieve system information?

System Information

- **Solution**: use system BIFs
- System information BIFs can be used to retrieve system information.
- They provide a good snapshot of the state of the processes.
- They can be used through a text based interface.
- No need to recompile the code.
- Do not affect the real time properties of the system.

System Information: Processes

processes()

BIF returning a list of all processes on the node

– Can be used in programs and in the Erlang shell

i()

Prints the list of processes with extra information:

– Pid and names,

– Initial and current module and function calls

– Heap size, stack size, reductions

– Number of unread messages

Can only be used from the shell

System Information: Processes

```
1> processes().
[<0.0.0>,<0.3.0>,<0.5.0>,<0.6.0>,<0.8.0>,<0.9.0>,<0.10.0>,
 <0.11.0>,<0.12.0>,<0.13.0>,<0.14.0>,<0.15.0>, ...,
 <0.23.0>,<0.24.0>,<0.25.0>,<0.26.0>,<0.27.0>,<0.31.0>]
2> i().
Pid           Initial Call          Heap     Reds Msgs
Registered    Current Function      Stack
<0.0.0>       otp_ring0:start/2     1597     2814    0
init          init:loop/1           2
<0.3.0>       erlang:apply/2        2584   196279    0
erl..._loader erl..._loader:loop/3  6
<0.5.0>       gen_event:init_it/6   610      226     0
error_logger  gen_event:fetch_msg/5 8
...
```

System information: Processes

- process_info(Pid [, Tag]).
- BIF returning one or all of the process attributes:
  - {initial_call, {M,F,A}}, {current_function, {M,F,A}}
  - {registered_name, Atom},
  - {error_handler, Module},
  - {trap_exit, true | false}, {links, Pids},
  - {status, waiting | running | runnable},
  - {messages, Msgs}, {message_queue_len, N},
  - {stack_size, Size}, {heap_size, Size}, {reductions, N},
  - etc.

System Information: Processes

```
1> process_info(whereis(error_logger)).
[{registered_name,error_logger},
 {current_function,{gen_event,fetch_msg,5}},
 {initial_call,{proc_lib,init_p,5}},
 {status,waiting}, {message_queue_len,0},
 {messages,[]},
 {links,[<0.0.0>,<0.23.0>]},
 {dictionary,[{'$ancestors',[<0.2.0>]}, ...}]},
 {trap_exit,true}, {error_handler,error_handler},
 {priority,normal}, {group_leader,<0.23.0>},
 {total_heap_size,987},
 {heap_size,610},
 {stack_size,8},
 ...
```

System Information: Registered Names

```
2> regs().
** Registered procs on node nonode@nohost **
Name             Pid       Initial Call         Reds Msgs
application_cont <0.6.0>   erlang:apply/2        436    0
code_server      <0.18.0>  erlang:apply/2     156896    0
erl_prim_loader  <0.3.0>   erlang:apply/2     197318    0
error_logger     <0.5.0>   gen_event:init_it/6   226    0
file_server_2    <0.17.0>  file_server:init/1     84    0
global_group     <0.16.0>  global_group:init/1    59    0
global_name_serv <0.12.0>  global:init/1          50    0
inet_db          <0.15.0>  inet_db:init/1        256    0
...
** Registered ports on node nonode@nohost **
Name                      Id          Command
```

System Information: Saving Calls

- **process_flag(save_calls, N)**
  process_info(Pid, last_calls)
- Activates call saving mode
- Saves N (1 .. 10,000) most recent:
  - Global function calls
  - BIF calls
  - Sending/receiving messages
- Has to be called by the process saving calls
- process_info/2 retrieves the calls

System Information: Saving Calls

```
1> process_flag(save_calls, 10).
0
2> Integer = 1234.
1234
3> process_info(self(), last_calls).
{last_calls,[{lists,reverse,1},   {erlang,self,0},
             {orddict,to_list,1}, {lists,foldl,3},
             {orddict,find,2},    {orddict,to_list,1},
             {lists,foldl,3},     {orddict,find,2},
             {lists,reverse,1},
             {erlang,process_info,2}]}
```

System Information: Saving Calls

- **erlang:process_display(Pid, backtrace).**
- Writes the information of the process on Standard Error
- It will display information on the
  - Stack
  - The Call Chain
- The most recent data is printed last
- **process_info(Pid, backtrace)** returns a binary containing a string with the information

System Information: Saving Calls

```
4> erlang:process_display(self(), backtrace).
Program counter: 0x0000000010668028 (unknown function)
CP: 0x0000000011480448 (erl_eval:do_apply/5 + 2328)

0x0000000012352da0 Return addr 0x0000000011457a70
(shell:exprs/7 + 680)
y(0)     [{'Integer',1234}]
y(1)     []
y(2)     none

0x0000000012352dc0 Return addr 0x0000000011457180
(shell:eval_exprs/7 + 144)
y(0)     []
...
```

System Information

- BIFs can be used to write meta-system programs.
- Poll the system to detect or break deadlocks.
- Poll thesystem to detect processes under heavy load.
- Analyse system performance.
- Gather useful data needed to solve hard to detect bugs.

Other Process Issues

- **erlang:suspend_process(Pid)**
  erlang:resume_process(Pid)
- Suspends and resumes a process
- Excellent to recreate timing-related bugs
- Should only be used for testing and debugging purposes.

Modules and Processes

- Modules
- System Information
  - Processes
  - Registered Names
  - Saving Calls
- Other Process Issues