

ngmoco:)

ESCRIPT

You were happily
playing poker with
your dogs...



...but they were really
wolves. Vicious,
hangry wolves.



THAT'S A GAME CHANGER, SON

ALSO, THEY ATE YOUR FACE OFF

BAM.

```
#!/usr/bin/env escript
```

```
main(_) ->
```

```
    io:format("Holy moly it's simple.~n", []),  
    erlang:halt(0).
```

[github.com/archaelus/
egc_examples](https://github.com/archaelus/egc_examples)

WHO?

- **Geoff Cant**
- Erlang hacker at ngmoco:)
- **archaelus** on irc / stackoverflow / erlang-questions / github
- Wellington -> Paris -> San Francisco

AGENDA

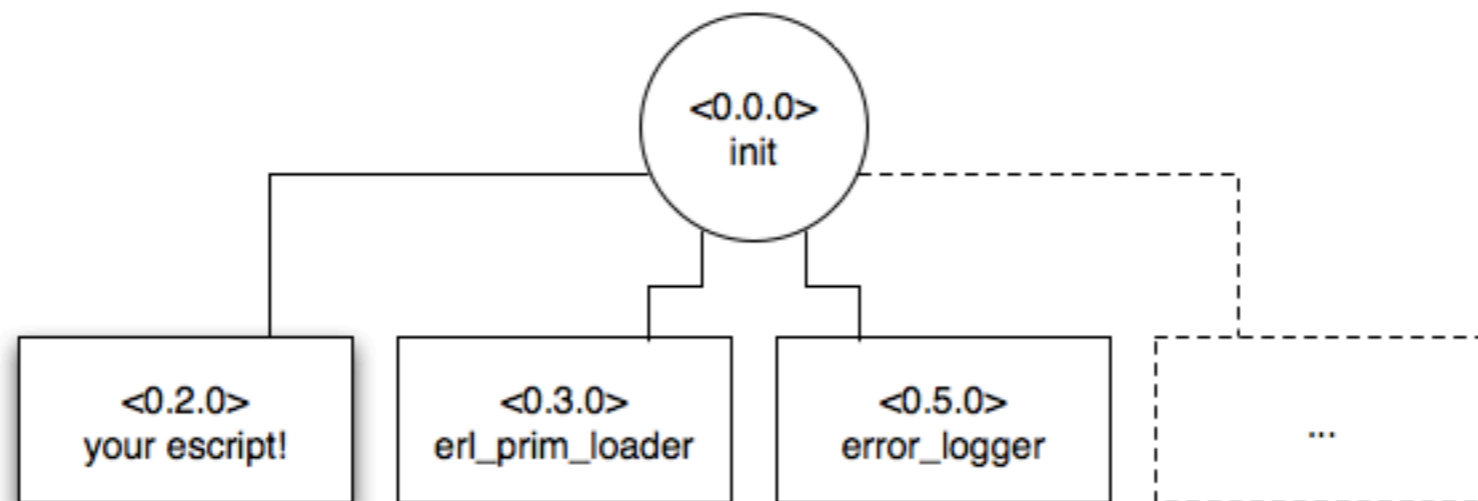
- ~~Magnets~~ Escript. How does it work?
- You said it would change the game.
- Show me some shenanigans!
- I can haz githubs?

EXPLAIN!

ESCRIPT ENVIRONMENT

- A full erlang node, but with a rapid boot sequence
- Interprets (or compiles) the escript text
- Calls `main([...])` with the string arguments to the escript
- Halts the node when the main function returns.

ESCRIPT ENVIRONMENT



INTERPRETED MODE

- Generated module name
- Closer to module code than Erlang shell code.

(?MODULE macro works!)

COMPILED MODE

- A bit faster than interpreted mode.
- Can use all standard compile options
- HiPE compile if you really want to!

BAM.

```
#!/usr/bin/env escript  
  
-mode(compile).  
  
main(_) ->  
    io:format("Holy moly it's simple.~n", []),  
    erlang:halt(0).
```

BAM.

```
#!/usr/bin/env escript  
  
-mode(compile).  
-compile(native).  
  
main(_) ->  
    io:format("Holy moly it's simple.~n", []),  
    erlang:halt(0).
```

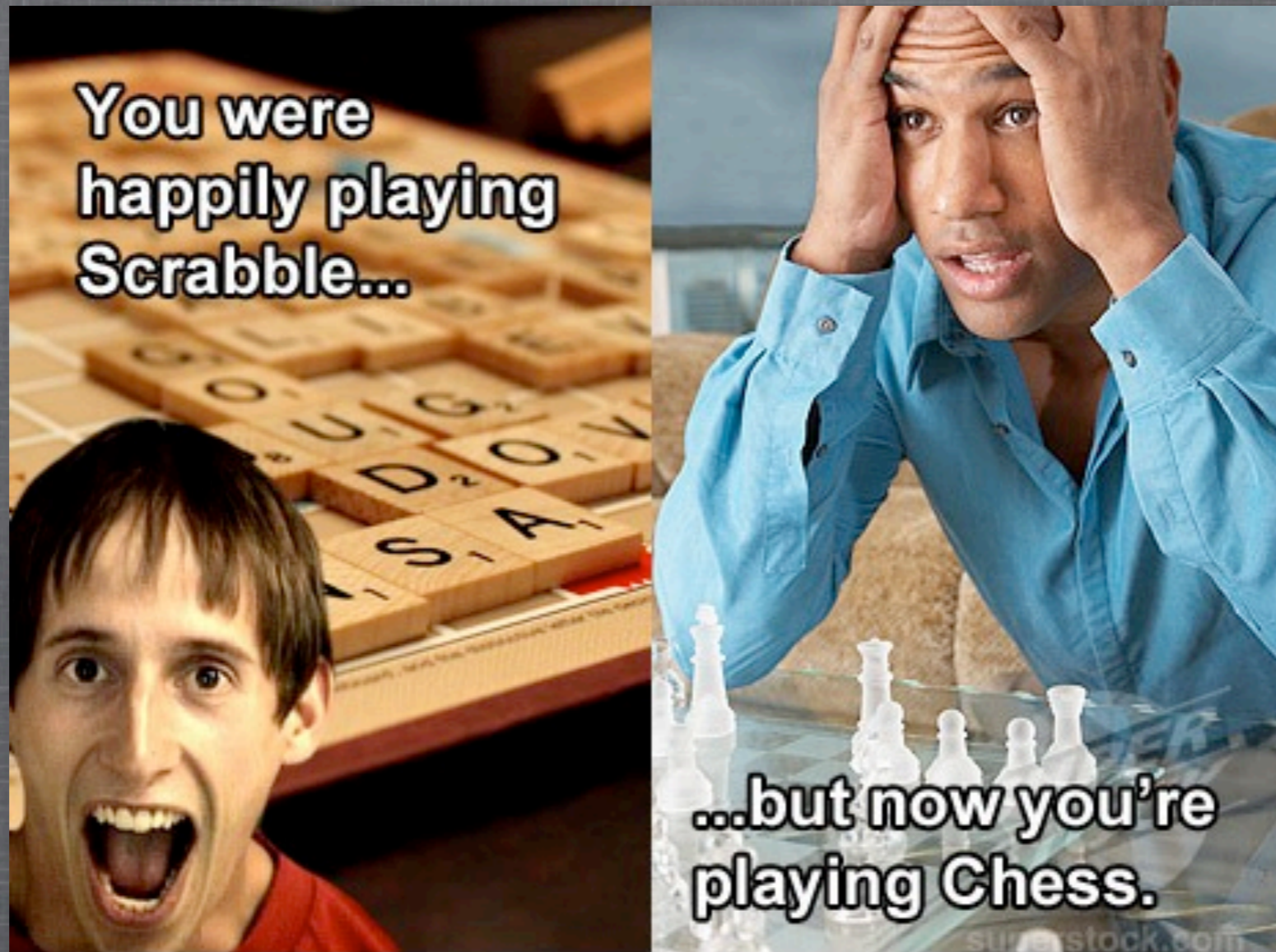
CODE ARCHIVES

- An escript can be:
 - an interpreted script
 - a compiled module
 - a zip file of compiled modules
- You can package an application or system as a (mostly) standalone executable

EXTRA GOODIES

- You can supply beam arguments
 - `%%! -escript main my_main_module`
 - `%%! -node soopah_script -setcookie blah`
- You don't have to create fancy escripts by hand
 - `escript:create /2`
- `escript -s`

CHANGE THE GAME!



**THAT'S A GAME
CHANGER, SON**

UNIX INTEGRATION

- Makefiles
- Rakefiles
- autotools

UNIX INTEGRATION

- init scripts
- Control scripts (reload config, ...)
- Health check scripts (nagios)
- System statistics (munin)

PREVIOUS OPTIONS

- Integrating with erlang is hard.
- Boot an erlang VM with a `_huge_` set of command line arguments
- Crazy RPC (REST interface? XML-RPC? ...)
- `erl_call` is pretty limited

ESCRIPT

- You start from a working Erlang environment
- No escaping or string interpolation problems
- Access to all kinds of information about the Erlang environment
- You get unix niceties

FETCH ME MY GAME CHANGING TROUSERS

THE GAME CHANGER
PERFORMANCE RIDESHORT
ORIGINAL TECHNOLOGY • GROUNDBREAKING INNOVATION

ZIPPIN'-OUT
REMOVEABLE RIDESHORT LINER

FLEX-LITE ULTRA
ANATOMICALLY FITTED NEOPRENE LINER

360 DEGREES STRETCH
FLEXIBLE OUTER NYLON SHELL

The advertisement features a person on the left wearing dark grey shorts with white vertical stripes and a white waistband. The shorts have 'THE GAME CHANGER' and 'ZIPPIN'-OUT' printed on them. On the right, a black background contains white and blue text and graphics. At the top is the brand name 'THE GAME CHANGER' with a logo, followed by 'PERFORMANCE RIDESHORT' and 'ORIGINAL TECHNOLOGY • GROUNDBREAKING INNOVATION'. Below this is a blue-bordered box with 'ZIPPIN'-OUT' in large blue letters and 'REMOVEABLE RIDESHORT LINER' in white. A graphic shows a hand zipping up the shorts. At the bottom, another blue-bordered box contains 'FLEX-LITE ULTRA' in blue and 'ANATOMICALLY FITTED NEOPRENE LINER' in white. Below that, a final blue-bordered box contains '360 DEGREES STRETCH' in blue and 'FLEXIBLE OUTER NYLON SHELL' in white. A watermark 'Text' is visible in the center.

DISTRIBUTION

- Start net_kernel
- Set a cookie
- Connect
- `rpc:go_hog_wild`

```

#!/usr/bin/env escript

-define(ME, filename:basename(escript:script_name())).

main(["ping", Node, Cookie]) ->
    net_kernel:start([my_name(), longnames]),
    erlang:set_cookie(my_name(), list_to_atom(Cookie)),
    Target = list_to_atom(Node),
    case net_adm:ping(Target) of
        pong ->
            io:format(rpc:call(Target, erlang, whereis, [user]),
                      "Hello from ~p.~n", [node()]),
            io:format("~s: ~s ~p~n", [?ME, Target, pong]),
            halt(0);
        Else ->
            io:format("~s: ~s ~p~n", [?ME, Node, Else]),
            halt(1)
    end;
main(_) ->
    io:format("~s: ping <Node> <Cookie>~n"
              "\tAttempt to contact the erlang node <Node>"
              " using the erlang distribution cookie <Cookie>.~n",
              [?ME]),
    halt(0).

my_name() ->
    Localhost = net_adm:localhost(),
    list_to_atom(?ME ++ "@" ++ Localhost).

```

SHENANIGANS!

REMOTE SHELL

- `%%! -noshell -noinput`
- `user_drv:start(['tty_sl -c -e',{Node,shell,start,[]}])`
- `trap_exit`, wait for exit.

```

#!/usr/bin/env escript
%%! -noshell -noinput
-define(ME, filename:basename(escript:script_name())).

main(["remsh", Node, Cookie]) ->
    net_kernel:start([my_name(), longnames]),
    erlang:set_cookie(my_name(), list_to_atom(Cookie)),
    Target = list_to_atom(Node),
    case net_adm:ping(Target) of
        pong ->
            process_flag(trap_exit, true),
            Shell = user_drv:start(['tty_sl -c -e', {Target, shell, start, []}]),
            true = erlang:link(Shell),
            io:format("Grabbed a remote shell on ~p~n.", [Target]),
            receive
                {'EXIT', Shell, _} -> ok
            end,
            halt(0);
        Else ->
            io:format("~s: ~s ~p~n", [?ME, Node, Else]),
            halt(1)
    end;
main(_) ->
    io:format("~s: remsh <Node> <Cookie>~n"
        "\tAttempt to start a remote shell on the erlang node <Node>"
        " using the erlang distribution cookie <Cookie>~n",
        [?ME]),
    halt(0).

my_name() ->
    Localhost = net_adm:localhost(),
    list_to_atom(?ME ++ "@" ++ Localhost).

```

ALRIGHT, FINE, I'LL
CHECK OUT ESCRIPT



RESOURCES

- <https://github.com/jcomellas/getopt>
- <https://github.com/basho/rebar>

ngmoco:)

We're hiring.