ArcBlock

Understanding Erlang Kernel

BROUGHT TO YOU BY

BOSHAN SUN

Why talk about Erlang Kernel?

Agenda

```
$ erl
Erlang/OTP 21 [erts-10.2.3] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1] [hipe] [dtrace]

Eshell V10.2.3  (abort with ^G)
1> length(erlang:processes()).
40
```

Not True

Live Demo

Agenda(real)

- How hello world works in Erlang
- How kernel application works

Processes

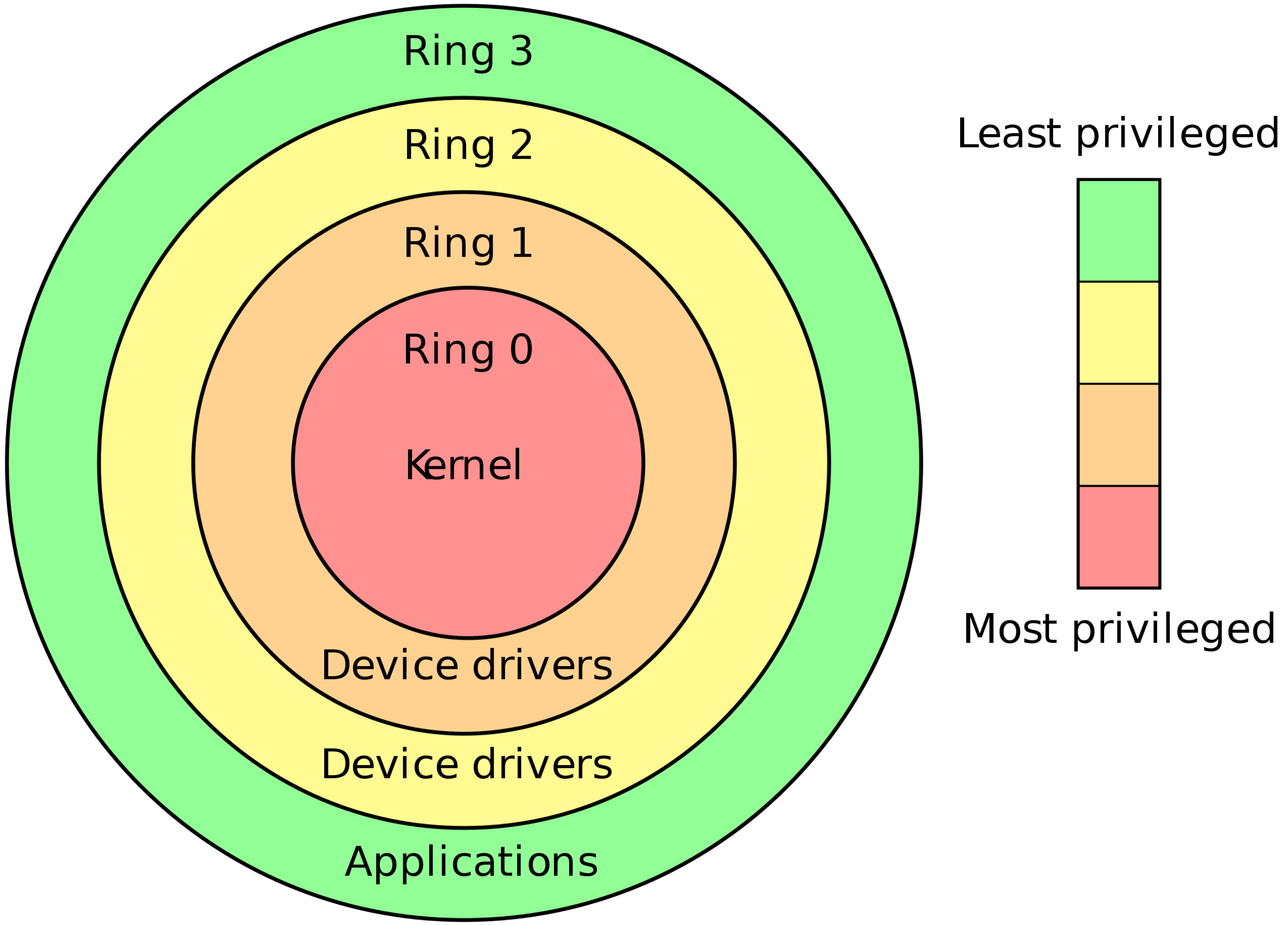
Registered Processes

```
$ erl
1> regs().
Name                Pid        Initial Call                 Reds Msgs
init                <0.0.0>    otp_ring0:start/2            1895    0
erts_code_purger    <0.1.0>    erts_code_purger:start/0       11    0
erl_prim_loader     <0.9.0>    erlang:apply/2              49425    0
logger              <0.41.0>   logger_server:init/1          469    0
application_controlle <0.43.0> erlang:apply/2               1229    0
kernel_sup          <0.48.0>   supervisor:kernel/1          2734    0
code_server         <0.49.0>   erlang:apply/2             118151    0
rex                 <0.51.0>   rpc:init/1                     70    0
global_name_server  <0.52.0>   global:init/1                  91    0
inet_db             <0.55.0>   inet_db:init/1                439    0
global_group        <0.56.0>   global_group:init/1           113    0
file_server_2       <0.57.0>   file_server:init/1            498    0
erl_signal_server   <0.58.0>   gen_event:init_it/6            63    0
standard_error_sup  <0.59.0>   supervisor_bridge:standar     136    0
standard_error      <0.60.0>   erlang:apply/2                 26    0
user_drv            <0.62.0>   user_drv:server/2            1848    0
user                <0.63.0>   group:server/3                125    0
kernel_refc         <0.66.0>   kernel_refc:init/1             78    0
kernel_safe_sup     <0.67.0>   supervisor:kernel/1           350    0
logger_sup          <0.68.0>   supervisor:logger_sup/1       290    0
logger_handler_watche <0.69.0> logger_handler_watcher:in      71    0
logger_std_h_default <0.71.0>  logger_h_common:init/1        337    0
disk_log_sup        <0.76.0>   supervisor:disk_log_sup/1     210    0
disk_log_server     <0.77.0>   disk_log_server:init/1        162    0
```

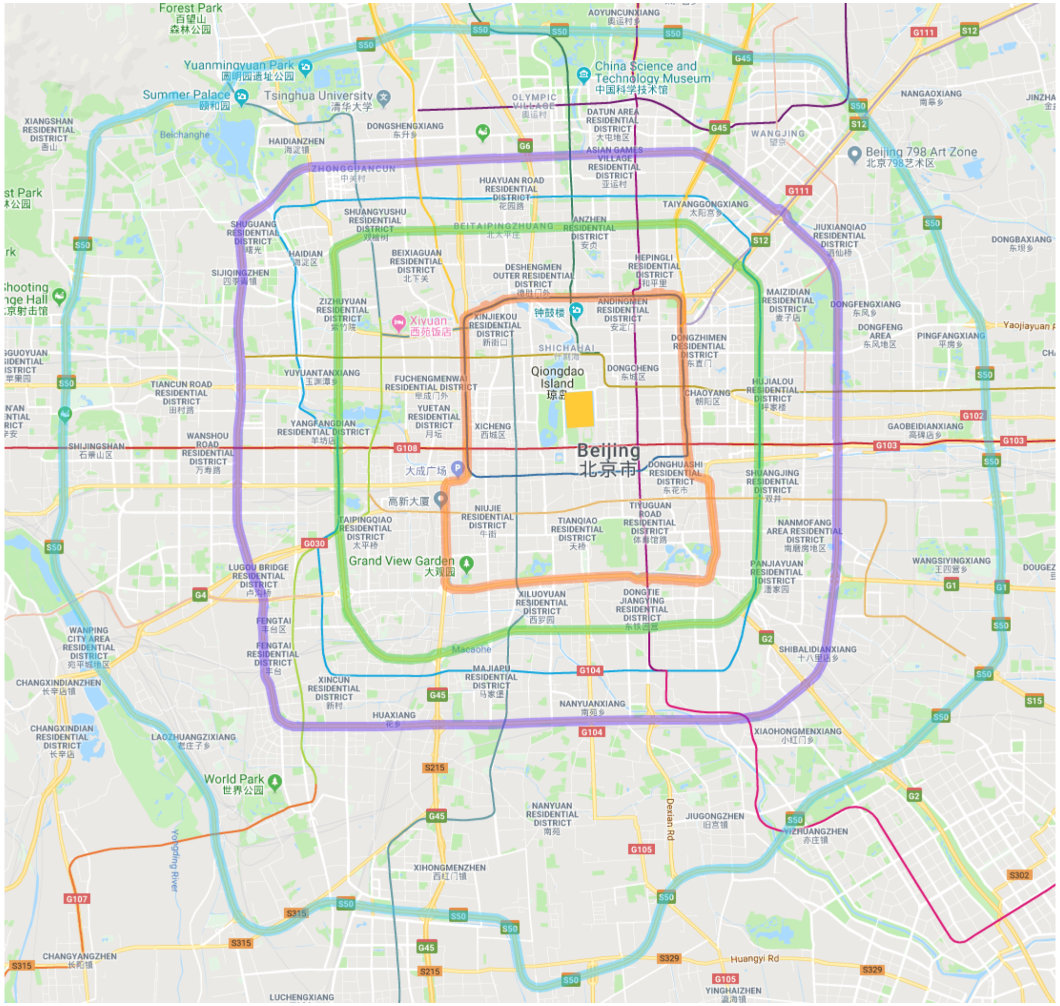
PID

```
<0.0.0>
   + -

xxxxxxxx xxxxxxxx xxxxxxxx xxxx0011
-------- -----+++ ++++++++ ++++0011

<0.1.0>
<0.2.0>
  ...
<0.32767.0>
<0.0.1>
<0.1.1>
  ...


1> spawn(fun() -> 1 end).
<0.85.0>
2> spawn(fun() -> 1 end).
<0.87.0>
3> spawn(fun() -> io:format("~p~n",[self()]) end),spawn(fun() -> io:format("~p~n",[self()]) end),ok.
<0.89.0>
<0.90.0>
ok
```

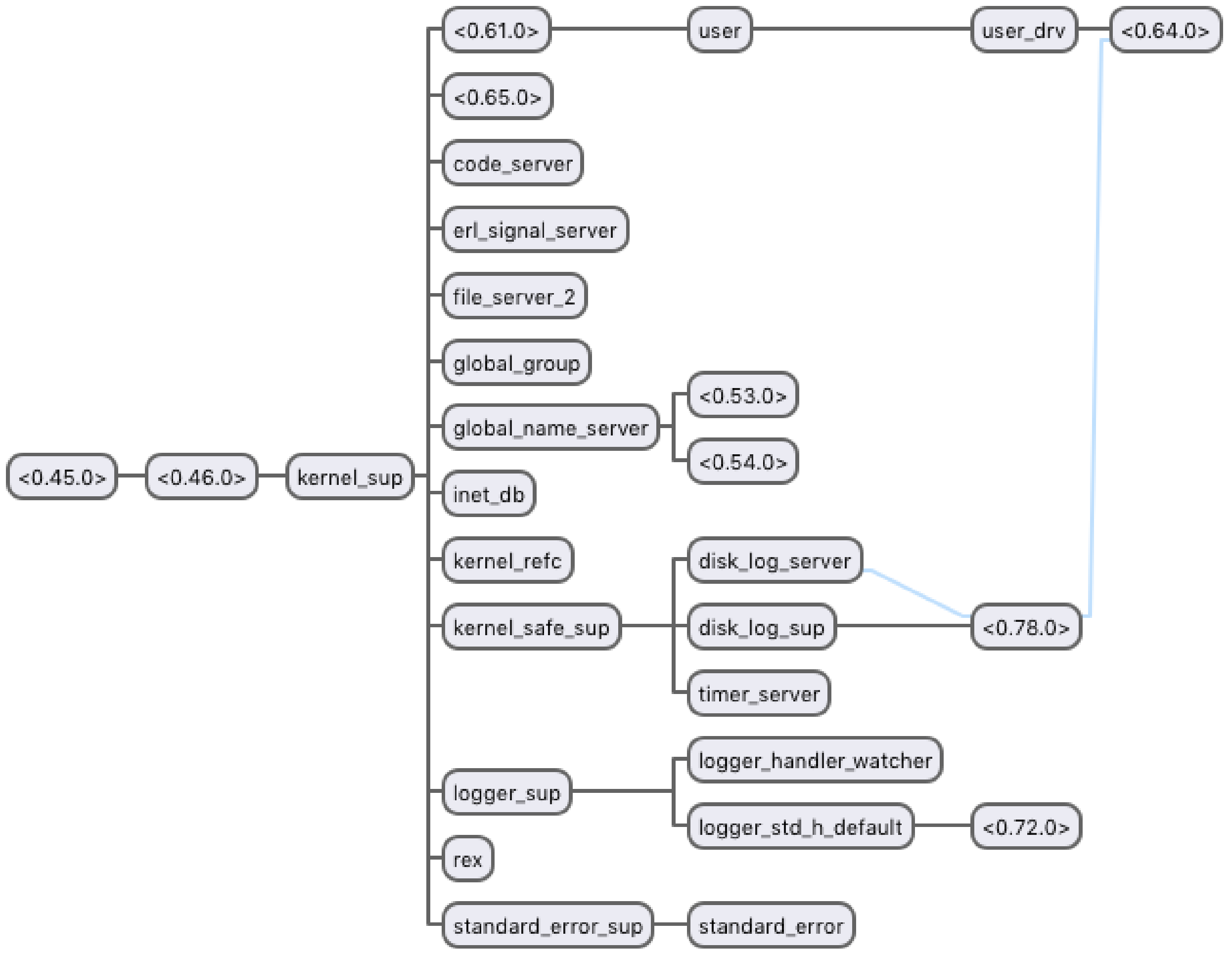
Protection Ring



Beijing Ring Roads



Kernel Application



boot script

```
$ erl
$ erl -boot start

$ ls /usr/local/Cellar/erlang/21.2.4/lib/erlang/bin/*.boot
/usr/local/Cellar/erlang/21.2.4/lib/erlang/bin/no_dot_erlang.boot
/usr/local/Cellar/erlang/21.2.4/lib/erlang/bin/start.boot
/usr/local/Cellar/erlang/21.2.4/lib/erlang/bin/start_clean.boot
/usr/local/Cellar/erlang/21.2.4/lib/erlang/bin/start_sasl.boot

$ ls /usr/local/Cellar/erlang/21.2.4/lib/erlang/bin/*.script
/usr/local/Cellar/erlang/21.2.4/lib/erlang/bin/start.script

$ erl -init_debug
{progress,preloaded}
{progress,kernel_load_completed}
{progress,modules_loaded}
{progress,init_kernel_started}
...
{progress,applications_loaded}
{progress,started}
Erlang/OTP 21 [erts-10.2.3] [source] [64-bit] [smp:12:12] [ds:12:12:10] [async-threads:1] [hipe] [dtrace]

Eshell V10.2.3  (abort with ^G)
1>
```

start mode

```
$ erl
$ erl -mode interactive
1> crypto:strong_rand_bytes(1).
<<"©">>

➜ erl -mode embedded
1> crypto:strong_rand_bytes(1).
** exception error: undefined function crypto:strong_rand_bytes/1
```

code_server

```
$ erl
1> code:get_mode().
interactive
2> code:all_loaded().
[{io,"/usr/local/Cellar/erlang/21.2.4/lib/erlang/lib/stdlib-3.7/ebin/io.beam"},
 {edlin,"/usr/local/Cellar/erlang/21.2.4/lib/erlang/lib/stdlib-3.7/ebin/edlin.beam"},
 ...]
3> code:get_path().
[".",
 "/usr/local/Cellar/erlang/21.2.4/lib/erlang/lib/kernel-6.2/ebin",
 "/usr/local/Cellar/erlang/21.2.4/lib/erlang/lib/stdlib-3.7/ebin",
 ...]
4> code:add_patha("/tmp").          % same as erl -pa tmp
true
5> code:get_path().
["/tmp",".",
 "/usr/local/Cellar/erlang/21.2.4/lib/erlang/lib/kernel-6.2/ebin",
 "/usr/local/Cellar/erlang/21.2.4/lib/erlang/lib/stdlib-3.7/ebin",
 ...]
```

code_server

```
$ iex
iex(1)> :code.get_path()
['/usr/local/Cellar/elixir/1.8.1/bin/../lib/elixir/ebin', ..., '.',
 '/usr/local/Cellar/erlang/21.2.4/lib/erlang/lib/kernel-6.2/ebin',
 '/usr/local/Cellar/erlang/21.2.4/lib/erlang/lib/stdlib-3.7/ebin',
 ...]

$ erl
1> 'Elixir.IO':puts(123).
** exception error: undefined function 'Elixir.IO':puts/1
2> code:add_patha("/usr/local/Cellar/elixir/1.8.1/bin/../lib/elixir/ebin").
true
3> 'Elixir.IO':puts(123).
123
ok

$ ERL_LIBS=/usr/local/Cellar/elixir/1.8.1/lib erl
1> 'Elixir.IO':puts(123).
123
ok
User switch command
 --> s 'Elixir.IEx'
 --> c 2
Interactive Elixir (1.8.1) - press Ctrl+C to exit (type h() ENTER for help)
iex(1)>
```

hello world

```
$ erl
1> io:format("hello world~n").
hello world
ok
2> spawn(fun() -> io:format("hello world~n") end).
hello world
<0.86.0>
3> [begin io:format("shell 1 ~p~n",[erlang:universaltime()]),timer:sleep(1000) end || _ <- lists:seq(1,1000)].
shell 1 {{2019,2,26},{2,5,45}}
shell 1 {{2019,2,26},{2,5,46}}
shell 1 {{2019,2,26},{2,5,47}}

User switch command      % ctrl + G
 --> s
 --> c
Eshell V10.2.3  (abort with ^G)
1> [begin io:format("shell 2 ~p~n",[erlang:universaltime()]),timer:sleep(1000) end || _ <- lists:seq(1,1000)].
shell 2 {{2019,2,26},{2,8,1}}
shell 2 {{2019,2,26},{2,8,2}}
shell 2 {{2019,2,26},{2,8,3}}

User switch command      % ctrl + G
 --> c 1
shell 1 {{2019,2,26},{2,8,8}}
shell 1 {{2019,2,26},{2,8,9}}
shell 1 {{2019,2,26},{2,8,10}}
```

Group leader

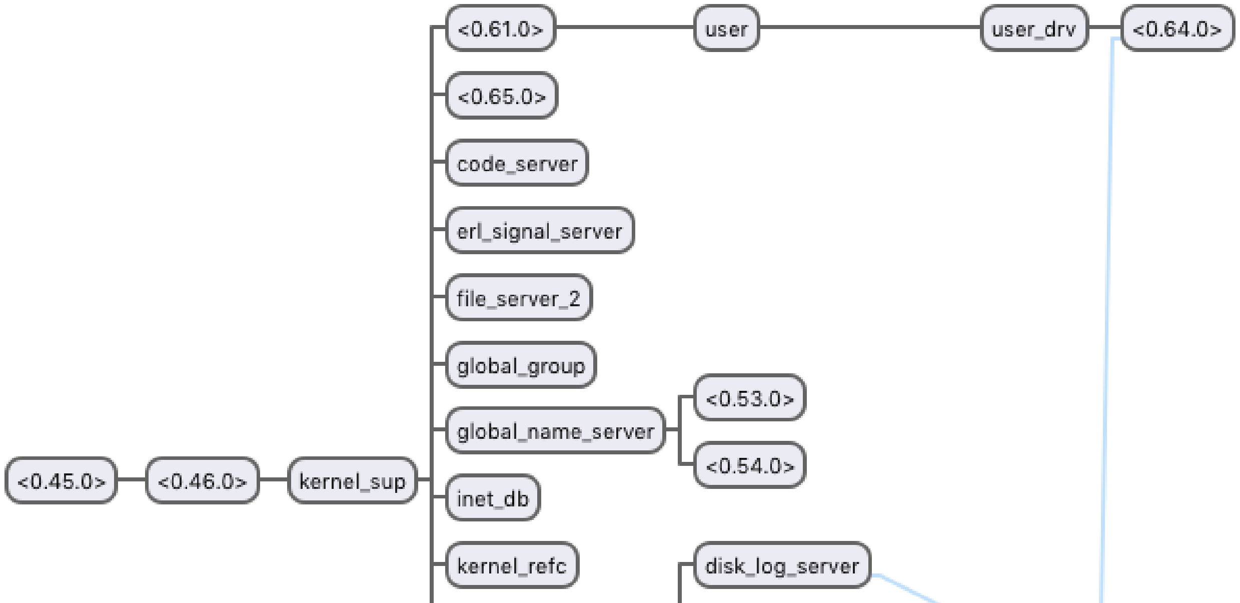
```
$ erl
1> group_leader().
<0.64.0>
2> group_leader() ! {io_request,self(),make_ref(),{put_chars,unicode,"hello\n"}}.
hello
{io_request,<0.83.0>,#Ref<0.3482152256.1555300357.130455>,
            {put_chars,unicode,"hello\n"}}
3> flush().
Shell got {io_reply,#Ref<0.3482152256.1555300357.130455>,ok}
ok
4> io:format("hello~n").
hello
ok
5> io:format(standard_io,"hello~n",[]).
hello
ok
6> io:format(group_leader(),"hello~n",[]).
hello
ok
7> io:format(user,"hello~n",[]).
hello
ok
7> io:format(init,"hello~n",[]).
hello
ok
```

user_drv

```
~

           {#Port<0.3>,{data,"1"}}                                {<0.62.0>,{data,"1"}}
  tty_sl   --------------------------> user_drv  ----------------------------------------------> group_leader()
#Port<0.3> <-------------------------- <0.62.0>  <----------------------------------------------   <0.64.0>
           {<0.62.0>,{command,[0|"1"]}}          {<0.64.0>,{requests,[{put_chars,unicode,"1"}]}}


$ erl
1> 1+1.
2
2> user_drv ! {#Port<0.3>,{data,"1+1.\r"}}.
{#Port<0.3>,{data,"1+1.\r"}}
3> 1+1.
2
4> user_drv ! {#Port<0.3>,{data,[$\^G]}}.

User switch command
 -->
```

Application Master

- Each application has an Application Master acting as a Group Leader
- Application master has a child process to synchronously start the root level supervisor and its children
- All I/O from processes within the application is send to the Application Master
- Then Application Master forward the I/O requests to the real group leader



Other

- logger/error_logger/disk_log
- gen_tcp/gen_udp/inet
- erl_signal_server
- file_server
- global/pg2
- heart
- rpc
- os

```
kernel_refc.erl
%%%-----------------------------------------------------------------
%%% This module implements a process that handles reference counters for
%%% various erts or other kernel resources which needs reference counting.
%%%
%%% Should not be documented nor used directly by user applications.
%%%-----------------------------------------------------------------
```

Chaos Monkey?

- Randomly killing processes is dangerous
- Shall only randomly kill processes belong to your application

Resources

- http://erlang.org/doc/apps/kernel/index.html
- https://github.com/erlang/otp/tree/master/erts/preloaded/src
- https://github.com/erlang/otp/tree/master/lib/kernel/src

Thank You

ArcBlock

www.arcblock.io