Assignment Project Exam Help

Message Passing

https://powcoder.com

CS511

Add WeChat powcoder

Message Passing

Exceptions

Links and Monitors

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Interaction Models

Assignment Project Exam Help

- Previously
  - Shared memory (low-level, non-structured)
  - Semaphores (low-level, non-structured)
  - Monitors (popular, structured, encapsulate synchronization)

https://powcoder.com

- So what's the problem with monitors?
  - Highly centralized (un/blocking processes, maintaining queues of blocked processes, encapsulating data)
  - For modern, distributed architectures, need for less centralized solution
  - Turn to interaction through communication rather than sharing

Add WeChat powcoder

The Message Passing Model

- No shared memory
  - A process sends a message
  - Another process receives the message
- Operations:

```
receive(Var);
send(PID,msg);
```

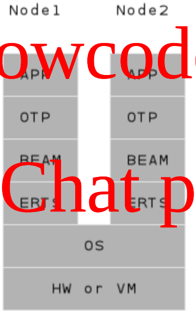
  - `receive` blocks until a message is available in the mailbox
  - `send(PID,msg)` is non-blocking; it sends message `msg` to process `PID`
- This model is the asynchronous communication model and is the one used in Erlang

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Nodes and Processes in Erlang[1]

- A distributed Erlang system consists of a number of Erlang runtime systems communicating with each other (instances of the VM)
- Each such runtime system is called a node

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

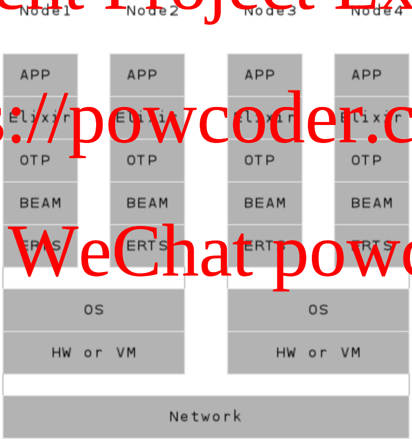


[1] Source: https://blog.stenmans.org/theBeamBook/

Distributed Nodes in Erlang

- A distributed Erlang system consists of a number of Erlang runtime systems communicating with each other (instances of the VM)

Assignment Project Exam Help



https://powcoder.com

Add WeChat powcoder

Nodes and Processes in Erlang

Assignment Project Exam Help

- Each such runtime system is called a node
  - node name is an atom `name@host`
    - `name` is the name given by the user
    - `host` is the full host name if long names are used, or the first part of the host name if short names are used

https://powcoder.com

- The name of a node may be consulted using `node()`

```
1> node().
nonode@nohost
```

Add WeChat powcoder

Processes and Communication in Erlang

- A process in a node has
  - a process id (pid)

```
1> self().
<0.78.0>
```

  - its own memory (a mailbox, a heap and a stack); and
  - a process control block (PCB) with information about the process.
- Message passing between processes at different nodes, as well as links and monitors, are transparent when pids are used
  - Registered names, however, are local to each node.
- Format of a PID:
  - node id where process lives; 0 if node is local
  - process index itelf (index into process table)
  - serial which increases every time MAXPROCS has been reached.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

A Simple Echo Server

- Process `echo` will receive a message and then send it back to the sender
- After that it will continue to wait for a new message
- It may be stopped by sending it the `stop` message

```
echo() ->
    receive
        {From, Msg} ->
            From ! {Msg},
            echo();
        stop -> true
    end.
```

- Processes are created using `spawn/1` and `spawn/3`

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

A Simple Echo Server (cont.)

```
-module(echo).
-export([start/0]).

echo() ->
    receive
        {From, Msg} ->
            From ! {Msg},
            echo();
        stop -> true
    end.

start() ->
    Pid = spawn(fun echo/0), % Returns pid of a new process
        % started by the application of echo/0 to []
    Token = "Hello Server!", % Sending tokens to the server
    Pid ! {self(), Token},
    io:format("Sent ~s~n",[Token]),
    receive
        {Msg} ->
            io:format("Received ~s~n", [Msg])
    end,
    Pid ! stop.    % Stop server
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

A Simple Echo Server

```
1> echo:start( ).
Sent Hello Server!
Received Hello Server!
stop
```

Assignment Project Exam Help

If we export `echo/0` we can spawn from the interpreter:

https://powcoder.com

```
59> X=spawn(fun echo:echo/0).
<0.198.0>
60> X!{self(),"hello"}.
{<0.60.0>,"hello"}
61> X.
<0.198.0>
```

Add WeChat powcoder

Note: the value of a send is the value of the message

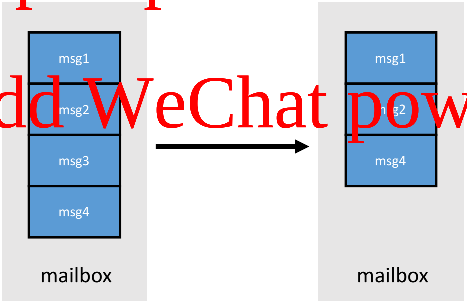
Reacting to Multiple Messages

- Erlang “listens” for messages from different senders
- In which order will they be processed?
- Can we force an order?
- A receive statement tries to find a match as early in the mailbox as it can

Assignment Project Exam Help

```erlang
receive
   msg3 -> 42
end
```

https://powcoder.com

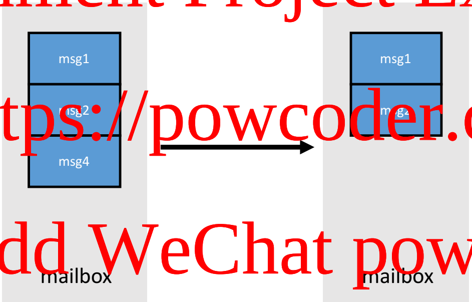


Add WeChat powcoder

Reacting to Multiple Messages

```
receive
    msg4 -> 42
end
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

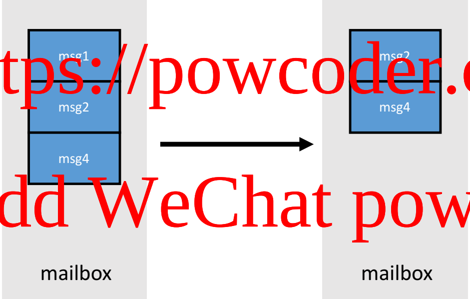


Reacting to Multiple Messages

Waiting for multiple messages

```
receive
    msg4 -> 42;
    _    -> 41
end
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

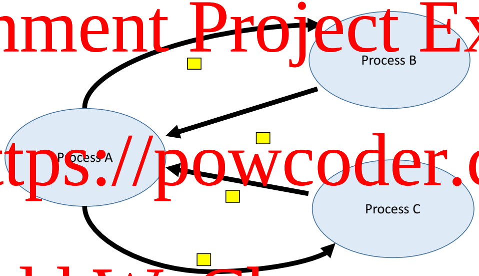


- The oldest message is tried against every pattern of the receive until one of them matches

Sources of Multiple Messages

Multiple messages can come from different processes

Assignment Project Exam Help



https://powcoder.com

Add WeChat powcoder

- How do we know who sent a message?
- Distinguish the source by Pids

Sources of Multiple Messages

```erlang
-module(echo2).
-export([start/0]).

echo() ->
    receive
     {From, Msg} ->
            timer:sleep(rand:uniform(100)),
            From ! {Msg},
            echo();
        stop ->
            true
    end.

% continued on next slide...
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- `timer:sleep(N)` sleeps a process for `N` milliseconds
- `rand:uniform(N)` produces a random integer between 1 and N

Sources of Multiple Messages

```
start() ->
    PidB = spawn(fun echo/0),
    PidC = spawn(fun echo/0),

    % sending tokens
    Token = 42,
    PidB ! {self(), Token},
    io:format("Sent ~w~n",[Token]),
    Token2 = 41,
    PidC ! {self(), Token2},
    io:format("Sent ~w~n",[Token2]),

    % receive message
    receive
        {Msg} ->
            io:format("Received ~w~n", [Msg])
    end,

    % stop echo-servers
    PidB ! stop,
    PidC ! stop.
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Sources of Multiple Messages

- How do we know who sent a message?
- Distinguish the source by Pids

Assignment Project Exam Help

```
-module(echo2).
-export([start/0]).

echo() ->
    receive
     {From, Msg} ->
            timer:sleep(rand:uniform(100)),
            From ! {self(), Msg},
            echo();
        stop ->
            true
    end.

% continued on next slide...
```

https://powcoder.com

Add WeChat powcoder

Sources of Multiple Messages

```
start() ->
    PidB = spawn(fun echo/0),
    PidC = spawn(fun echo/0),

    % sending tokens
    Token = 42,
    PidB ! {self(), Token},
    io:format("Sent ~w~n",[Token]),
    Token2 = 41,
    PidC ! {self(), Token2},
    io:format("Sent ~w~n",[Token2]),

    % receive messages
    receive
      {PidB, Msg} ->
            io:format("Received from B: ~w~n", [Msg]);
      {PidC, Msg} ->
            io:format("Received from C: ~w~n", [Msg])
    end,

    % stop echo-servers
    PidB ! stop,
    PidC ! stop.
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Sources of Multiple Messages

```
11> echo2:start().
Sent42
Sent41
Received from B: 42
stop
12> echo2:start().
Sent42
Sent41
Received from B: 42
stop
13> echo2:start().
Sent42
Sent41
Received from C: 41
stop
14> echo2:start().
Sent42
Sent41
Received from B: 42
stop
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Sources of Multiple Messages

Multiple messages can come from the same processes

Assignment Project Exam Help

- Send several messages of the same shape and continue computing
- When receiving the responses, how can the code match them to the appropriate request?

https://powcoder.com

- BIF `make_ref` provides globally unique reference objects (references for short) different from every other object in the Erlang system including remote nodes

Add WeChat powcoder

- References can be used to uniquely identify messages

Sources of Multiple Messages

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
-module(echo3).
-export([start/0]).

echo() ->
    receive
        {From, Ref, Msg} ->
            From ! {self(), Ref, Msg},
            echo();
        stop ->
            true
    end.

% continues in next slide...
```

Sources of Multiple Messages

```
start() ->
    PidB = spawn(fun echo/0),
    % sending tokens
    Token = 42,
    Ref = make_ref(),
    PidB ! {self(), Ref, Token},
    io:format("Sent~w~n",[Token]),
    Token2 = 41,
    Ref2 = make_ref(),
    PidB ! {self(), Ref2, Token2},
    io:format("Sent~w~n",[Token2]),
    % receive messages
    receive
        {PidB, Ref2, Msg} ->
            io:format("Received 41? ~w~n", [Msg]);
        {PidB, Ref, Msg} ->
            io:format("Received 42? ~w~n", [Msg])

    end,

    % stop echo-servers
    PidB ! stop.
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Selective Receive

Assignment Project Exam Help

- Clauses can have guards
- Guards must be composed of terminating functions (BIFs)

https://powcoder.com

```
receive
  {Pid, Ref, N} when N>0 -> ...
```

Add WeChat powcoder

Timeouts

```
f(Pid) ->
    receive
        {Pid, Msg} -> Msg
    after 3000 ->
        timeout
end.
```

Assignment Project Exam Help

- The after part will be triggered if 3000 milliseconds have passed without receiving a message that matches the pattern.
- Other uses

https://powcoder.com

```
sleep(T) ->
    receive
    after T ->
        ok
end.

flush() ->
    receive
        _ -> flush()
    after 0 ->
        ok
end.
```

Add WeChat powcoder

Exercise

- Implement a semaphore
- Use the `when` clause

Assignment Project Exam Help

Template that you can start from:

https://powcoder.com

```
-module(semaphore).
-compile(export_all).

make_semaphore(Permits) ->
spawn(?MODULE,semaphore,[Permits]).

% complete
```

Add WeChat powcoder

- `?MODULE`: macro that refers to the name of the current module

A Semaphore

```erlang
-module(semaphore).
-compile(export_all).

make_semaphore(Permits) ->
    spawn(?MODULE,semaphore,[Permits]).

semaphore(0) ->
    receive
        {From,Ref,release} ->
            semaphore(1)
    end;
semaphore(P) when P>0 ->
    receive
        {From,Ref,release} ->
            From!{self(),Ref,ok},
            semaphore(P+1);
        {From,Ref,acquire} ->
            From!{self(),Ref,ok},
            semaphore(P-1)
    end.
```

- `semaphore` could be specified as a FSM

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Semaphore - Print “a” before “b”

```
start() ->
    S = make_semaphore(0),
    spawn(?MODULE,p1,[S]),
    spawn(?MODULE,p2,[S]).

release(S) -> % could be included in semaphore module
    R = make_ref(),
    S!{self(),R,release},
    receive
        {S,R,ok} ->
            done
    end.

p1(S) ->
    io:format("a"),
    release(S).

p2(S) ->  %  acquire is inlined
    R = make_ref(),
    S!{self(),R,acquire},
    receive
        {S,R,ok} ->
            io:format("b")
    end.
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Assignment Project Exam Help

Message Passing

Exceptions

https://powcoder.com

Links and Monitors

Add WeChat powcoder

Three Kinds of Exceptions

- Errors
  - Ends the execution in the current process and includes a stack trace of the last functions
  - Errors are the means for a function to stop its execution when you can't expect the calling code to handle what just happened
- Throws
  - Used for cases that the programmer can be expected to handle (`try...catch`).
- Exits
  - Same as errors except used to signal abnormal termination between processes.
  - More lightweight than errors in that stack trace not included

Note: `try...catch` actually can catch them all

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Errors – Example

Assignment Project Exam Help

```
1> erlang:error(badarith).
** exception error: bad argument in an arithmetic expression
2> erlang:error(custom_error).
** exception error: custom_error
3> catch(1+a).
{'EXIT',{badarith,[{erlang,'+',[1,a],[]},
      {erl_eval,do_apply,6,[{file,"erl_eval.erl"},{line,681}]},
      {erl_eval,expr,5,[{file,"erl_eval.erl"},{line,434}]},
      {shell,exprs,7,[{file,"shell.erl"},{line,686}]},
      {shell,eval_exprs,7,[{file,"shell.erl"},{line,642}]},
      {shell,eval_loop,3,[{file,"shell.erl"},{line,627}]}]}}
```

https://powcoder.com

Add WeChat powcoder

Assignment Project Exam Help

Message Passing

https://powcoder.com

Exceptions

Links and Monitors

Add WeChat powcoder

Links

- Pid1 can be linked to Pid2 by calling `link(Pid2)`
  - Creates a two-way link
- Terminating processes emit exit signals to all linked processes, which can terminate as well or handle the exit in some way.
- This feature can be used to build hierarchical program structures where some processes are supervising other processes, for example, restarting them if they terminate abnormally.

Note: Some comments on monitors are present at the end of these set of slides

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Example

```
-module(linkmon).
-compile(export_all).

myproc() ->
    timer:sleep(2000),
    exit(reason).
```

In the shell:

```
> c(linkmon).
{ok,linkmon}
> self().
<0.79.0>
> spawn(fun linkmon:myproc/0).
<0.75.0>
> self().
<0.79.0>
> link(spawn(fun linkmon:myproc/0)).
true
** exception error: reason
> self().
<0.83.0>
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Another Example

```
chain(0) ->
  receive
    _ -> ok
  after 2000 ->
    exit("chain dies here")
  end;

chain(N) ->
  Pid = spawn(fun() -> chain(N-1) end),
  link(Pid),
  receive
    _ -> ok
  end.
```

In the shell:

```
1> c(linkmon).
{ok,linkmon}
2> link(spawn(linkmon, chain, [3])).
true
** exception error: "chain dies here"
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Another Example (cont.)

Assignment Project Exam Help

```
[shell] == [3] == [2] == [1] == [0]
[shell] == [3] == [2] == [1] == *dead*
[shell] == [3] == [2] == *dead*
[shell] == [3] == *dead*
[shell] == *dead*
*dead, error message shown*
[shell] <-- restarted
```

https://powcoder.com

- After the process running `linkmon:chain(0)` dies, the error is propagated down the chain of links until the shell process itself dies because of it.
- The crash could have happened in any of the linked processes
  - because links are bidirectional, you only need one of them to die for the others to follow suit.

Add WeChat powcoder

On Number of Links and Linking

Assignment Project Exam Help

- Links cannot be stacked.
  - Calling `link/1` multiple times for the same two processes, will still create only one link between them
  - A single call to `unlink/1` will be enough to tear it down.

https://powcoder.com

- `link(spawn(Function))` or `link(spawn(M,F,A))` happens in more than one step. In some cases, it is possible for a process to die before the link has been set up and then provoke unexpected behavior.

Add WeChat powcoder

  - `spawn_link/1-3` spawns and links as an atomic operation

Trapping Exit Signals

Assignment Project Exam Help

- In order to be reliable, an application needs to be able to both kill and restart a process quickly.
  - Links convenient for the killing part but restarting is missing.

https://powcoder.com

- When a linked process terminates, it terminates with an exit reason that is sent through a special message known as an exit signal
  - Eg. exit signal with exit reason "chain dies here"

Add WeChat powcoder

```
exit("chain dies here")
```

Trapping Exit Signals

Assignment Project Exam Help

- The default behaviour when a process receives an exit signal with an exit reason other than normal, is to terminate and in turn emit exit signals with the same exit reason to its linked processes.

https://powcoder.com

- System processes: normal processes, except they can convert exit signals to regular messages.
  - Done by calling `process_flag(trap_exit, true)` in a running process.

Add WeChat powcoder

- Allows a process to react to exit signals

Chain Example Revisited

Chain example with a system process at the beginning

```
1> process_flag(trap_exit, true).
true
2> spawn_link(fun() -> linkmon:chain(3) end).
<0.49.0>
3> receive X -> X end.
{'EXIT',<0.49.0>,"chain dies here"}
```

Description of behavior:

```
[shell] == [3] == [2] == [1] == [0]
[shell] == [3] == [2] == [1] == *dead*
[shell] == [3] == [2] == *dead*
[shell] == [3] == *dead*
[shell] <-- {'EXIT,Pid,"chain dies here"} -- *dead*
[shell] <-- still alive!
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

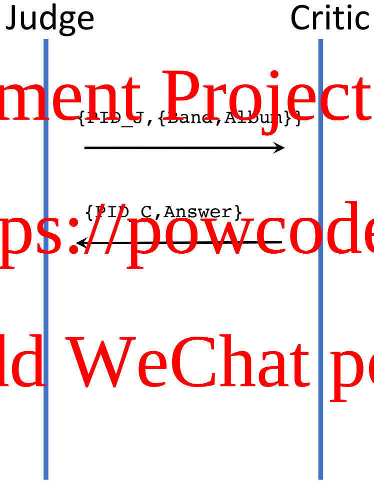
Kill Reason

- Acts as a special signal that can't be trapped.
- Ensures any process you terminate with it will be dead.
  - A last resort, when everything else has failed.
- As the kill reason can never be trapped, it needs to be changed to `killed` when other processes receive the message.
  - Otherwise, every other process linked to it would in turn die for the same kill reason and would in turn kill its neighbors, and so on.
  - This explains why `exit(kill)` looks like `killed` when received from another linked process.

```
> spawn_link(fun() -> exit(kill) end).
** exception exit: killed
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

MSC for Critic Example



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Restarting Processes

```
start_critic() ->
    spawn(?MODULE, critic, []).

judge(Pid, Band, Album) ->
  Pid ! {self(), {Band, Album}},
  receive
    {Pid, Criticism} -> Criticism
    after 2000 ->
      timeout
  end.

critic() ->
  receive
    {From, {"Rage Against the Turing Machine", "Unit Testify"}} ->
      From ! {self(), "They are great!"};
    {From, {"System of a Downtime", "Memoize"}} ->
      From ! {self(), "They're not Johnny Crash but they're good."
    {From, {"Johnny Crash", "The Token Ring of Fire"}} ->
      From ! {self(), "Simply incredible."};
   {From, {_Band, _Album}} ->
      From ! {self(), "They are terrible!"}
  end,
  critic().
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Restarting Processes

Assignment Project Exam Help

```
1> c(linkmon).
{ok,linkmon}
2> Critic = linkmon:start_critic().
<0.47.0>
3> linkmon:judge(Critic, "Genesis", "The Lambda Lies Down on Broad
"They are terrible!"
```

https://powcoder.com

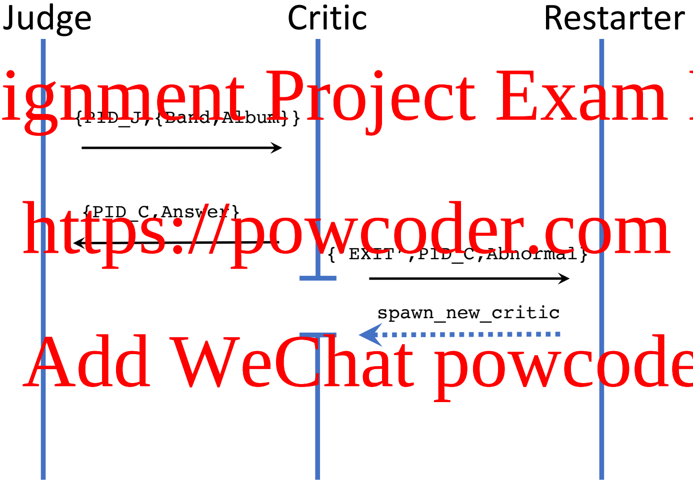
We now kill the `Critic` process

```
4> exit(Critic, solar_storm).
true
5> linkmon:judge(Critic, "Genesis", "A trick of the Tail Recursion
timeout
```

Add WeChat powcoder

We need a “supervisor” process to keep critics alive

MSC for Critic Example



Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Restarting Processes

```
start_critic2() ->
spawn(?MODULE, restarter, []).

restarter() ->
  process_flag(trap_exit, true),
  Pid = spawn_link(?MODULE, critic, []),
  receive
    {'EXIT', Pid, normal} -> % not a crash
      ok;
    {'EXIT', Pid, shutdown} -> % manual termination, not a crash
      ok;
    {'EXIT', Pid, _} ->
  restarter()
end.
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Problem: Pid of the critic is part of internal state, it is not known

```
1> c(linkmon).
{ok,linkmon}
2> linkmon:start_critic2().
<0.48.0>
3> linkmon:judge(?????, "Genesis", "The Lambda Lies Down on Broadw
```

Restarting Processes

- We can name a process, using an atom, rather than use its pid via `erlang:register/2`
- If a process dies, it will automatically lose its name or you can also use `unregister/1`
- You can get a list of all registered processes with `registered/0` or a more detailed one with the shell command `regs()`.

```erlang
restarter() ->
  process_flag(trap_exit, true),
  Pid = spawn_link(?MODULE, critic, []),
  register(critic, Pid),
  receive
    {'EXIT', Pid, normal} -> % not a crash
      ok;
    {'EXIT', Pid, shutdown} -> % manual termination, not a crash
      ok;
    {'EXIT', Pid, _} ->
      restarter()
  end.
```

What about the judge?

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Restarting a Process

Assignment Project Exam Help

```
judge2(Band, Album) ->
  critic ! {self(), {Band, Album}},
  Pid = whereis(critic),
  receive
    {Pid, Criticism} -> Criticism
  after 2000 ->
    timeout
  end.
```

https://powcoder.com

Add WeChat powcoder

Restarting a Process

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

```
1> linkmon:start_critic2().
<0.58.0>
2> whereis(critic).
<0.59.0>
3> linkmon:judge2("Genesis", "A trick of the Tail Recursion").
"They are terrible!"
4> exit(whereis(critic),solar_storm).
true
5> linkmon:judge2("Genesis", "A trick of the Tail Recursion").
"They are terrible!"
6> whereis(critic).
<0.63.0>
```

Race Conditions due to Shared State

Assignment Project Exam Help

- `critic` is stored in a shared registry
- There are processes that read it such as `judge2`
- And processes that write to it such as `restarter`
- Race conditions are therefore possible

https://powcoder.com

Add WeChat powcoder

Race Conditions due to Shared State – Example 1

1. critic ! Message
2. critic receives
3. critic replies
4. critic dies
5. whereis fails
6. critic is restarted
7. code crashes

```
judge2(Band, Album) ->
  critic ! {self(), {Band, Album}},
  %% critic dies at this point
  %% register still not updated
  Pid = whereis(critic), %% fails (returns undefined)
  receive
    {Pid, Criticism} -> Criticism  %% undefined!=Pid
  after 2000 ->
    timeout
  end.
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Race Conditions due to Shared State – Example 2

1. `critic ! Message`
2. critic receives
3. critic replies
4. critic dies
5. critic is restarted
6. whereis picks up wrong pid
7. message never matches

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

```
judge2(Band, Album) ->
  critic ! {self(), {Band, Album}},
  %% critic dies at this point
  %% register updated with new Pid
  Pid = whereis(critic), %% successful (but different Pid)
  receive
    {Pid, Criticism} -> Criticism %% no match
  after 2000 ->
    timeout
  end.
```

- Both may be solved by replacing the use of `whereis` (and Pid matching) to that of reference matching

Adding References to Messages

```
judge2(Band, Album) ->
  Ref = make_ref(),
  critic ! {self(), Ref, {Band, Album}},
  receive
    {Ref, Criticism} -> Criticism
  after 2000 ->
    timeout
  end.

critic2() ->
  receive
    {From, Ref, {"Rage Against the Turing Machine", "Unit Testify"
      From ! {Ref, "They are great!"};
    {From, Ref, {"System of a Downtime", "Memoize"}} ->
      From ! {Ref, "They're not Johnny Crash but they're good."};
    {From, Ref, {"Johnny Crash", "The Token Ring of Fire"}} ->
      From ! {Ref, "Simply incredible."};
    {From, Ref, {_Band, _Album}} ->
      From ! {Ref, "They are terrible!"}
  end,
  critic2().
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Assignment Project Exam Help

Appendix: More on Exceptions

https://powcoder.com

Appendix: Monitors

Add WeChat powcoder

Revisiting Exceptions – How Processes Trap Them

- `spawn_link(fun() ->ok end)`
  - Untrapped Result: Nothing
  - Trapped Result: `{'EXIT', <0.61.0>, normal}`
  - The process exited normally, without a problem.
- `spawn_link(fun() ->exit(reason) end)`
  - Untrapped Result: `** exception exit: reason`
  - Trapped Result: `{'EXIT', <0.55.0>, reason}`
  - The process has terminated for a custom reason.
- `spawn_link(fun() ->exit(normal) end)`
  - Untrapped Result: Nothing
  - Trapped Result: `{'EXIT', <0.58.0>, normal}`
  - Emulates process terminating normally.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Revisiting Exceptions

- `spawn_link(fun() ->1/0 end)`
  - Untrapped Result:

    `Error in process <0.44.0> with exit value: {badarith, [{erlang, '/',`
  - Trapped Result:

    `{'EXIT', <0.52.0>, {badarith, [{erlang, '/', [1,0]}]}}`
- `spawn_link(fun() ->erlang:error(reason) end)`
  - Untrapped Result:

    `Error in process <0.47.0> with exit value: {reason, [{erlang, apply,`
  - Trapped Result:

    `{'EXIT', <0.74.0>, {reason, [{erlang, apply, 2}]}}}`
  - Similar to 1/0.
- `spawn_link(fun() ->throw(rocks) end)`
  - Untrapped Result:

    `Error in process <0.51.0> with exit value: {{nocatch, rocks}, [{erlan`
  - Trapped Result:

    `{'EXIT', <0.79.0>, {{nocatch, rocks}, [{erlang, apply, 2}]}}`
  - Because the throw is never caught by a try ... catch, it bubbles up into an error, which in turn bubbles up into an EXIT. Without trapping exit, the process fails.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Revisiting Exceptions – the `exit/2` case

Allows a process to kill another one from a distance, safely

Assignment Project Exam Help

- `exit(self(), normal)`
  - Untrapped Result: `** exception exit: normal`
  - Trapped Result: `{'EXIT', <0.31.0>, normal}`
  - When not trapping exits, `exit(self(), normal)` acts the same as `exit(normal)`.

https://powcoder.com

- `exit(spawn_link(fun() ->timer:sleep(50000) end), normal)`
  - Untrapped Result: nothing
  - Trapped Result: nothing

Add WeChat powcoder

- `exit(spawn_link(fun() ->timer:sleep(50000) end), reason)`
  - Untrapped Result: `** exception exit: reason`
  - Trapped Result: `{'EXIT', <0.52.0>, reason}`

Revisiting Exceptions – the `exit/2` case

Assignment Project Exam Help

- `exit(spawn_link(fun() ->timer:sleep(50000) end), kill)`
  - Untrapped Result: `** exception exit: killed`
  - Trapped Result: `{'EXIT', <0.58.0>, killed}`
- `exit(self(), kill)`
  - Untrapped Result: `** exception exit: killed`
  - Trapped Result: `** exception exit: killed`
- `spawn_link(fun() ->exit(kill) end)`
  - Untrapped Result: `** exception exit: killed`
  - Trapped Result: `{'EXIT', <0.67.0>, kill}`

https://powcoder.com

Add WeChat powcoder

Monitors

Assignment Project Exam Help

- Special type of link with two differences
  - they are unidirectional,
  - can monitor via a registered name, and
  - they can be stacked.

https://powcoder.com

- Allows a process to, unobtrusively, monitor another one
- Useful for when you have multiple libraries that you call and they all need to know whether a process is alive or not
  - You can stack links and remove them individually

Add WeChat powcoder

Example

`erlang:monitor/2` sets up a monitor, where the first argument is the atom process and the second one is the pid

Assignment Project Exam Help

```
1> erlang:monitor(process, spawn(fun() -> timer:sleep(500) end)).
#Ref<0.0.0.77>
2> flush().
Shell got {'DOWN',#Ref<0.0.0.77>,process,<0.63.0>,normal}
ok
```

https://powcoder.com

- When monitored process goes down, send message to monitor: `{'DOWN', MonitorReference, process, Pid, Reason}`.
- The reference allows you to demonitor the process.
  - Monitors are stackable, so it's possible to take more than one down.
  - References allow you to track each of them in a unique manner.

Add WeChat powcoder

Example

Atomic function to spawn process while monitoring it:

```
3> {Pid, Ref} = spawn_monitor(fun() -> receive _ -> exit(boom) end
{<0.73.0>,#Ref<0.0.0.100>}
4> erlang:demonitor(Ref).
true
5> Pid ! die
die
6> flush().
ok
```

- We demonitored the other process before it crashed hence no trace of it dying.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder