References


Erlang Documentation

Reading

Ways a Process can Die

Normal
    Process code ends

Exceptions
    error, throw, exit

exit(kill)
    Very lethal

Murder
    exit(Pid, Reason) ends process with Pid
Linking Processes

Bi-directional Links

Pid = spawn_link(fun() -> ... end)

Links current process with new process

link(APid)

Links current process with process with APid

One-way Link

erlang:monitor(process, Pid)
if Pid dies current process is notified
Links and Death

Let processes A & B be linked

If either process dies unnaturally it sends an exit signal to the other process

An exit signal kills the process it is sent to

   Except if the process is a system process

   An exit signal puts a message in the process mailbox

However if exit(kill) was the original cause of death the other process will die even if it is a system process (but examples don't support this)
Crime Does Pay

Let processes A & B be linked

If A kills B using exit(Pid, Reason) A does not die

If B kills A using exit(Pid, Reason) B does not die
System Process

A process becomes a system process by calling the function

process_flag(trap_exit,true)
Example

-module (exitTests).
-export ([start/1]).

start (Reason) ->
    A = spawn(fun() -> a() end),
    B = spawn(fun() -> b(A, Reason) end).

a () ->
    process_flag(trap_exit,true),
    read(a).

b (Parent,Reason) ->
    link(Parent),
    case Reason of
        normal -> true;
        error -> erlang:error(raiseError);
        throw -> throw(raiseThrow);
        exit -> exit(exit);
        kill -> exit(kill)
    end.

read (Pid) ->
    io:format("Read For ~p ~n", [Pid]),
    receive
        Any ->
            io:format("Pid ~p received ~p~n", [Pid,Any]),
            read(Pid)
        after 1000 ->
            io:format("Process ~p time out~n", [Pid])
    end.
Example

15> exitTests:start(normal).
Read For a
<0.146.0>
Pid a received {'EXIT',<0.146.0>,normal}
Read For a
Process a time out

16> exitTests:start(throw).
Read For a
<0.152.0>
Pid a received {'EXIT',<0.152.0>,{{nocatch,raiseThrow},[[exitTests,b,2]]}}

=ERROR REPORT==== 10-Sep-2008::21:36:05 ===
Error in process <0.152.0> with exit value: {{nocatch,raiseThrow},[[exitTests,b,2]]}

Read For a
Example

16> exitTests:start(kill).
Read For a
<0.149.0>
Pid a received {'EXIT',<0.149.0>,kill}
Read For a
Process a time out

This does not match what the book indicates should happen.
I don't care if a Process dies

Pid = spawn(fun() -> ... end)

I want to die if a process I create dies

Pid = spawn_link(fun() -> ... end)

I want to handle errors if a process I create dies

process_flag(trap_exit, true),
Pid = spawn_link((fun() -> ... end),
....
receive
    {'EXIT', SomePid, Reason} ->
    %handle the problem
    ...