

# Morte por concorrência

---

Roberto Alves Neto



```
import java.lang.*;

class MiniThread extends Thread {
    int n;

    MiniThread(int m) {
        n = m;
    }

    public void run() {
        do {
            yield();
            n--;
        } while (n > 0);
    }
}
```

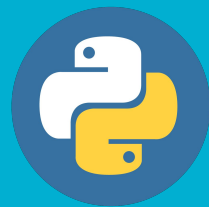
```
public class App {
    public static void main(String[] args) throws Exception {
        int threads = 100;
        int times = 10;

        long tempoInicial = System.currentTimeMillis();

        for (int i = threads; i > 0; i--) {
            MiniThread t = new MiniThread(times);
            t.start();
        }

        long tempoFinal = System.currentTimeMillis();

        System.out.printf("%.3f ms%n", (tempoFinal - tempoInicial) / 1000d);
    }
}
```



```
from threading import Thread
import time

def mini_thread(times):
    for i in range(times):
        time.sleep(0)

def death(n, m):
    threads = n
    times = m
    for i in range(threads):
        thread = Thread(target=mini_thread, args=[times])
        thread.start()
```

```
• • •  
-module(ring).  
-export([send/2]).
```

```
%% @doc Send M messages through a ring of N processes.
```

```
send(M, N) ->  
    statistics(runtime),  
    H = lists:foldl(  
        fun(Id, Pid) -> spawn_link(fun() -> loop(Id, Pid, M) end) end,  
        self(),  
        lists:seq(N, 2, -1)),  
    {_ , Time} = statistics(runtime),  
    io:format("~p processes spawned in ~p ms~n", [N, Time]),  
    statistics(runtime),  
    H ! M,  
    loop(1, H, M).
```

```
loop(Id, Pid, M) ->  
    receive  
        1 ->  
            {_ , Time} = statistics(runtime),  
            io:format("~p messages sent in ~p ms~n", [M, Time]),  
            exit(self(), ok);  
        Index ->  
            erlang:yield(),  
            Pid ! Index - 1,  
            loop(Id, Pid, M)  
    end.
```



# Comparação tempos

---

	100 P e 100 M	1000 P e 1000 M	10000 P e 1000 M	100000 P e 10000 M
Java	0,007 s	0,014 s	0,225 s	2,342 s
Python	0,059 s	5,48 s	61,82 s	+10 min. Desisti
Erlang	0,001 s	0.020 s	0,123 s	0,530 s