

# Lua multiple concurrent processes

Asked 12 years, 8 months ago   Modified 12 years, 8 months ago   Viewed 6k times



I want to execute multiple processes concurrently from a lua script e.g.

2

```
os.execute("cmd1")  
os.execute("cmd2")  
os.execute("cmd3")
```



where cmd1,2 and 3 are continually running processes. When i do this as above, cmd2 and 3 will only run when cmd1 is finished. Any idea on this? Should i be using "fork" or something equivalent?



Thanks in advance

concurrency

process

lua

Share Edit Follow Flag

asked Feb 28, 2012 at 10:34



[greatodensraven](#)

301

1

7

15

4 Answers

Sorted by:

[Reset to default](#)

Date modified (newest first)



(answer mostly copied from [Call popen with environment](#))

3

There is an `os.spawn` function in the [ExtensionProposal](#) API.

You can use it as follows:

```
require"ex"
local proc, err = os.spawn{
    command = e.."/bin/aprogr",
    args = {
        "arg1",
        "arg2",
        -- etc
    },
    env = {
        A = 100, -- I assume it toStrings the value
        B = "Hi",
        C = "Test",
    },
    -- you can also specify stdin, stdout, and stderr
    -- see the proposal page for more info
}
if not proc then
    error("Failed to aprogrinate! "..tostring(err))
end



-- if you want to wait for the process to finish:
local exitcode = proc:wait()
```

[lua-ex-pai](#) provides implementations for POSIX and Windows. It allows the spawning of multiple concurrent processes.

You can find precompiled binaries of this implementation bundled with the [LuaForWindows](#) distribution.

Here is a more concise version of your use case:

```
require"ex"
local cmd1_out = io.pipe()
local cmd1_proc = assert(os.spawn("cmd", {
    stdout = cmd1_out,
}))
local cmd2_out = io.pipe()
local cmd2_proc = assert(os.spawn("cmd", {
    stdout = cmd1_out,
}))
-- perform actions with cmd1 and cmd2
```

Share Edit Follow Flag  
Does lua-ex-api still maintained? I dont understand google code. Looks odd and old. Any hint on this? – [Ismoh](#) May 23, 2022 at 8:42  
edited May 23, 2017 at 10:34  
answered Feb 29, 2012 at 1:21  
 Community Bot  
1 1  
 [Deco](#)  
5,149 1 17 18



Try simply adding & at the end of your commands:

2

```
os.execute("cmd1 &")  
os.execute("cmd2 &")  
os.execute("cmd3 &")
```



This should work on an operative system. On windows there might be a way to to the same, but I have no idea of what it is.



Share Edit Follow Flag

answered Feb 28, 2012 at 13:42



[kikito](#)

52.5k 33 153 195



You've got several solutions to your problem:

5

1. Depending on your operating system shell, you might use & to put tasks into the background. For example: `os.execute('(sleep 10&& echo bar) & echo foo')`
2. [Lua Posix](#) comes with a `posix.fork()` function
3. [Lua Lanes](#) gives you multithreading in Lua, so you might be able to just use `os.execute` in separate lanes (note that 'threads' in a Lua context usually refers to coroutines instead of native OS threads).



Share Edit Follow Flag

answered Feb 28, 2012 at 10:46



[jpjacobs](#)

9,529 37 45



It's a old post yea, but does 1. work on windows? – [Ismoh](#) May 23, 2022 at 8:41

1



Yes and no. `os.execute` will work, but the command string has to be changed to something windows-shell specific. – [jpjacobs](#) Aug 6, 2022 at 18:24

**2**

That's because Lua is single threaded. To run it concurrently you'll need to provide a multi-threaded solution for Lua (not coroutines, because they're microthreads!), like [lua pthreads](#).

[Share](#) [Edit](#) [Follow](#) [Flag](#)

answered Feb 28, 2012 at 10:46

[Kornel Kisielewicz](#)**57.3k** 15 112 149