parallel

A simple way to run several functions at once.

Functions are not actually executed simultaneously, but rather this API will automatically switch between them whenever they yield (e.g. whenever they call **`coroutine.yield`**, or functions that call that - such as **`os.pullEvent`** - or functions that call that, etc - basically, anything that causes the function to "pause").

Each function executed in "parallel" gets its own copy of the event queue, and so "event consuming" functions (again, mostly anything that causes the script to pause - eg **`os.sleep`**, **`rednet.receive`**, most of the **`turtle`** API, etc) can safely be used in one without affecting the event queue accessed by the other.

> ⚠ **WARNING**
>
> When using this API, be careful to pass the functions you want to run in parallel, and *not* the result of calling those functions.
>
> For instance, the following is correct:
>
> ```
> local function do_sleep() sleep(1) end
> parallel.waitForAny(do_sleep, rednet.receive)
> ```
> Run ▷
>
> but the following is **NOT**:
>
> ```
> local function do_sleep() sleep(1) end
> parallel.waitForAny(do_sleep(), rednet.receive)
> ```
> Run ▷

Changes

- **New in version 1.2**

`waitForAny(...)`	Switches between execution of the functions, until any of them finishes.
`waitForAll(...)`	Switches between execution of the functions, until all of them are finished.

§ `waitForAny(...)` [Source]

Switches between execution of the functions, until any of them finishes. If any of the functions errors, the message is propagated upwards from the `parallel.waitForAny` call.

Parameters

1. `...` : `function` The functions this task will run

Usage

- Print a message every second until the `q` key is pressed.

```lua
local function tick()
    while true do
        os.sleep(1)
        print("Tick")
    end
end
local function wait_for_q()
    repeat
        local _, key = os.pullEvent("key")
    until key == keys.q
    print("Q was pressed!")
end

parallel.waitForAny(tick, wait_for_q)
print("Everything done!")
```

Run ▷

§ `waitForAll(...)` [Source]

Switches between execution of the functions, until all of them are finished. If any of the functions errors, the message is propagated upwards from the `parallel.waitForAll` call.

Parameters

1. `...` : `function` The functions this task will run

Usage

- Start off two timers and wait for them both to run.

```lua
local function a()
    os.sleep(1)
    print("A is done")
end
```

Run ▷

```lua
local function b()
    os.sleep(3)
    print("B is done")
end

parallel.waitForAll(a, b)
print("Everything done!")
```

Last updated on 2025-07-06