This page is for historical reference.

"Multiprocessing" feature (done)

Desirable Skills:

- Experience with concurrency and interprocess communication
- Strong experience with C
- Familiar with VimL (Vim script) and Vim concepts (quickfix list, buffers, etc.)

Description:

p2p architecture for data sharing between multiple Nvim instances. Similar to Python's multiprocessing module, the idea is to to offload Nvim tasks (VimL and/or Lua) to child Nvim processes.

Here's a picture of the potential workflow:

1. parent calls `invoke_async(Foo)`
2. parent spawns new child `nvim` process
3. parent sends command name `Foo` + state to child
4. parent does other, unrelated work
5. child completes its `Foo` work
6. child sends notification (method name + state) to parent

Difficulty: Hard

Code license: Apache 2.0

Mentor: Justin M Keyes (@justinmk)

TUI (Terminal UI) remote attachment (done)

Desirable Skills:

C

Description:

The built-in UI is called the TUI. It looks like Vim, but internally it is decoupled from the UI and "screen" layout subsystem. It was designed to be able to connect to other (remote) instances of Nvim, but this hasn't been implemented yet. #7438

Nvim is both a server and a client. Nvim (client) can connect to any other Nvim (server). And Nvim GUIs can show the screen of a remote Nvim server.

But the built-in Nvim TUI cannot show the screen of a remote Nvim server. That's the goal of this project.

- It is not "live share". It's just showing the remote UI in a client TUI.
- Networking question are out of scope. It is assumed the client has an SSH tunnel or named pipe to connect to.

The Nvim API model is:

- `:help api`: general API functions
- `:help ui`: UI events
- Any client can call any API function.
- If a client calls `nvim_ui_attach`, then it is a "UI client". This simply means that Nvim will send UI events as msgpack-rpc notifications on the channel.

That's how *every* Nvim UI works. And that's how the TUI client (this project proposal) will work.

Python "demo UI" may be helpful: https://github.com/neovim/python-gui

The simplest UI is the "fake UI" implemented in `test/functional/ui/screen.lua` from the Nvim test suite. It creates a text UI from real Nvim UI events. This allows us to write Lua tests that check the state of the UI, by simply writing the text UI in the test. In the Neoivm repo, "git grep 'screen:expect' " shows all of the places where this is used.

The `example_spec.lua` test shows a simple example. You can try it by running this shell command:

```
TEST_FILE=test/functional/example_spec.lua make functionaltest
```

Overview of the `screen.lua` "fake UI" implementation:

- `Screen._wait()` / `Screen:sleep()` runs the event loop to consume UI events
- `Screen:_redraw()` dispatches UI events to the appropriate handlers
- For example `Screen:_handle_grid_line()` consumes a line event, and updates some tables (self._grids and self._attr_table).
  - And those tables are literally the contents of the fake UI that `Screen:expect()` tests against.

Use cases:

- Connect to any Nvim. Unlike tmux, Nvim UI client can connect to any other running Nvim, including GUIs.
- Potential for using `libnvim` as the RPC core of any Nvim API client, to eliminate the need for clients to implement their own msgpack-RPC handling.

Expected Result:

Implement a TUI "remote UI" client. Modify the TUI subsystem so that it can display a remote Nvim instance. The C codebase already has msgpack support, an event-loop, and the ability to connect to sockets/named pipes/etc.

- Extend `tui/tui.c` to:
  1. connect to a channel
  2. get UI events from the channel
  3. unpack the events and call the appropriate handlers
- Extend `tui/input.c` to:
  1. send user input to the channel (i.e. call the nvim_input() API function)

Example:

```
nvim --servername foo
```

will connect to the Nvim server at address `foo`. The `nvim` client instance sends input to the remote Nvim server, and reflects the UI of the remote Nvim server. So the `nvim` client acts like any other external (G)UI.

Difficulty: Medium

Code license: Apache 2.0

Mentor: Justin M Keyes (@justinmk), Björn Linse (@bfredl)