



i3: the tree branch

an introduction to what's coming

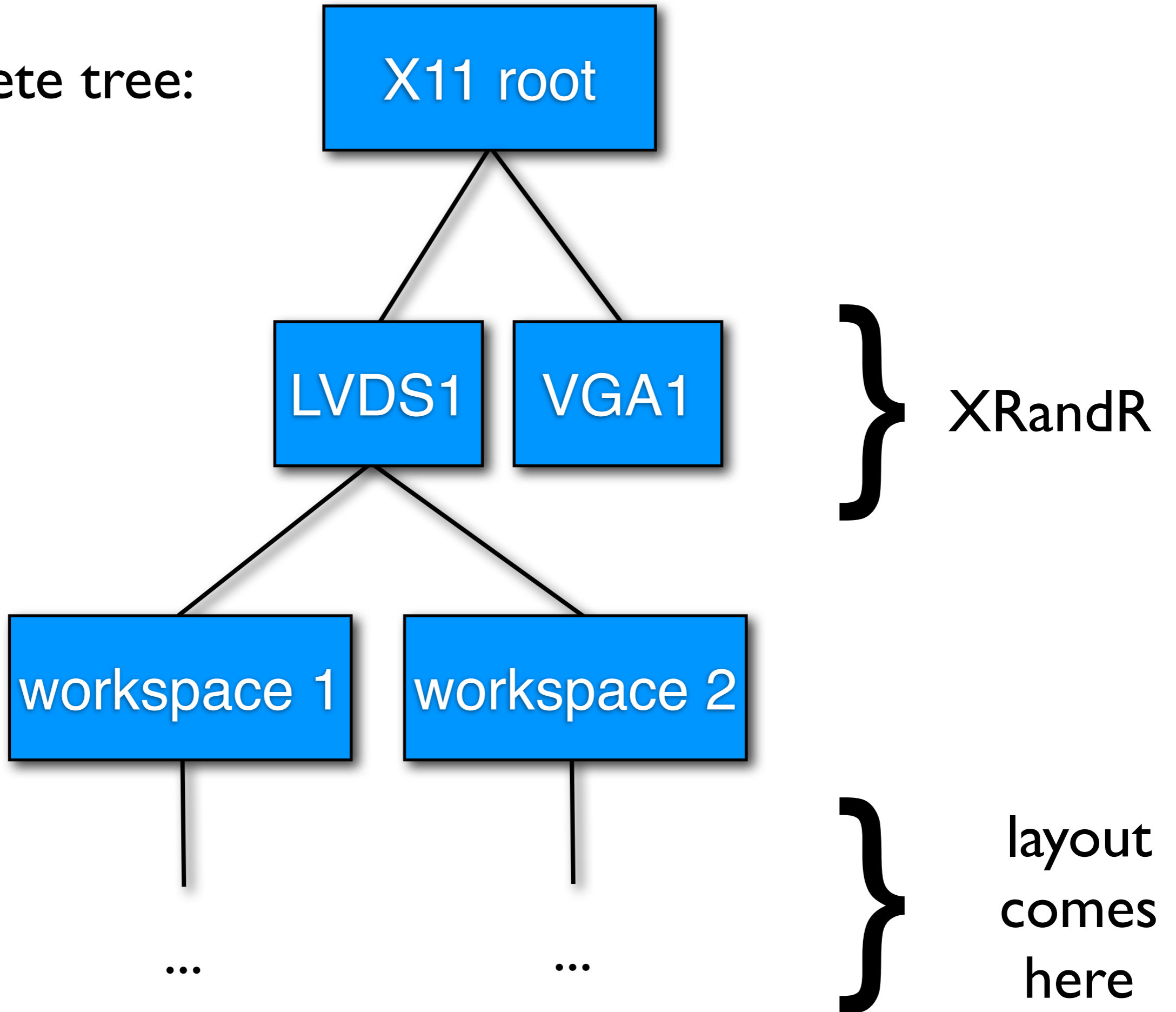
Disclaimer

- You need to know the current version of i3, this is not a presentation for beginners.
- The colors inside the screenshots look strange because it is a debugging version.

What is the tree branch?

- major refactoring of i3 (still in progress)
- easier data structures (good for devs and users)
- layout is tree-based now
- tree exposed via IPC (in JSON format):
much more test-driven development

a complete tree:



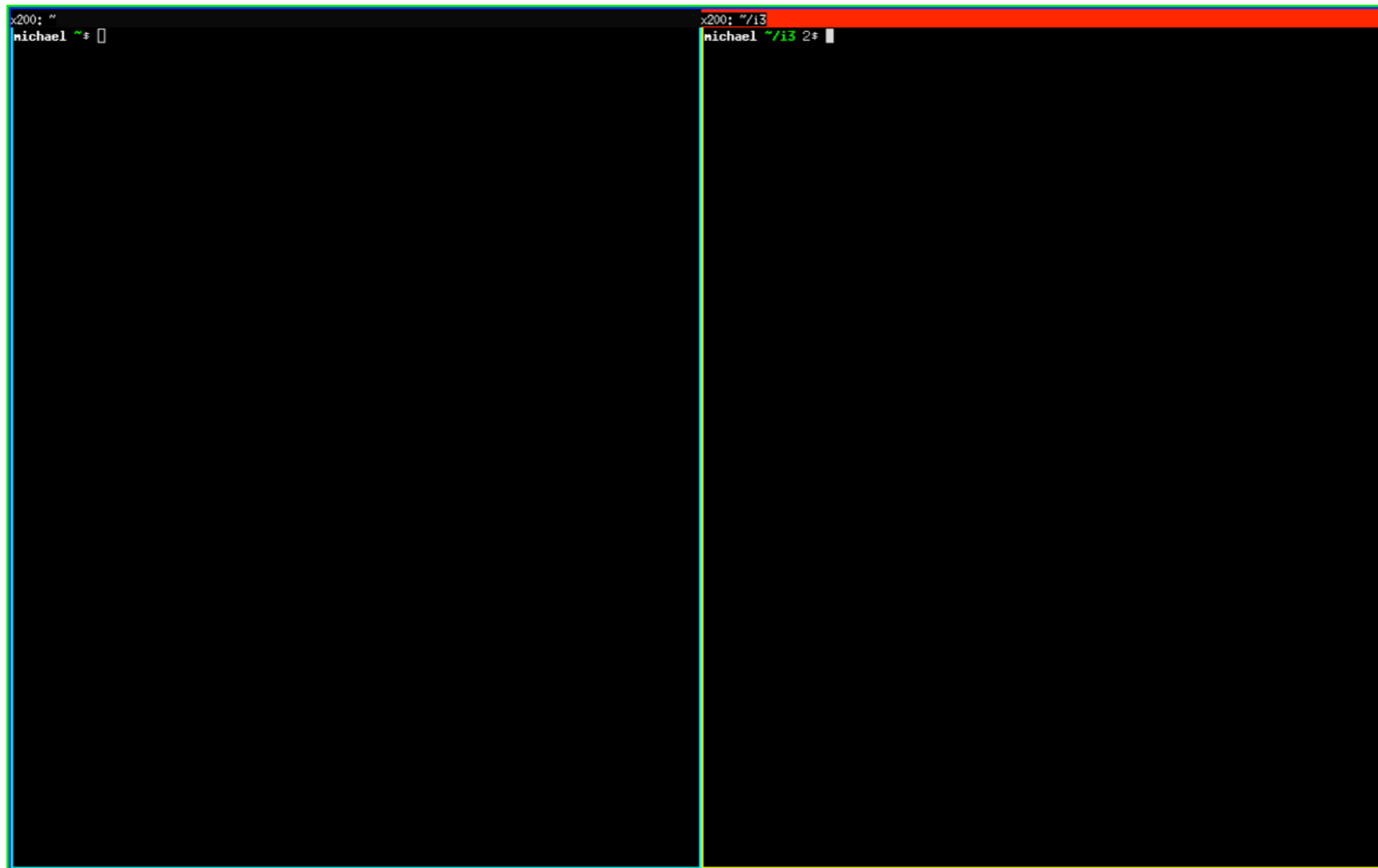
2: basic window management

NEW: The table metaphor is gone.

(It was too complicated and had too many ugly side-effects)

So, how do you move windows around?

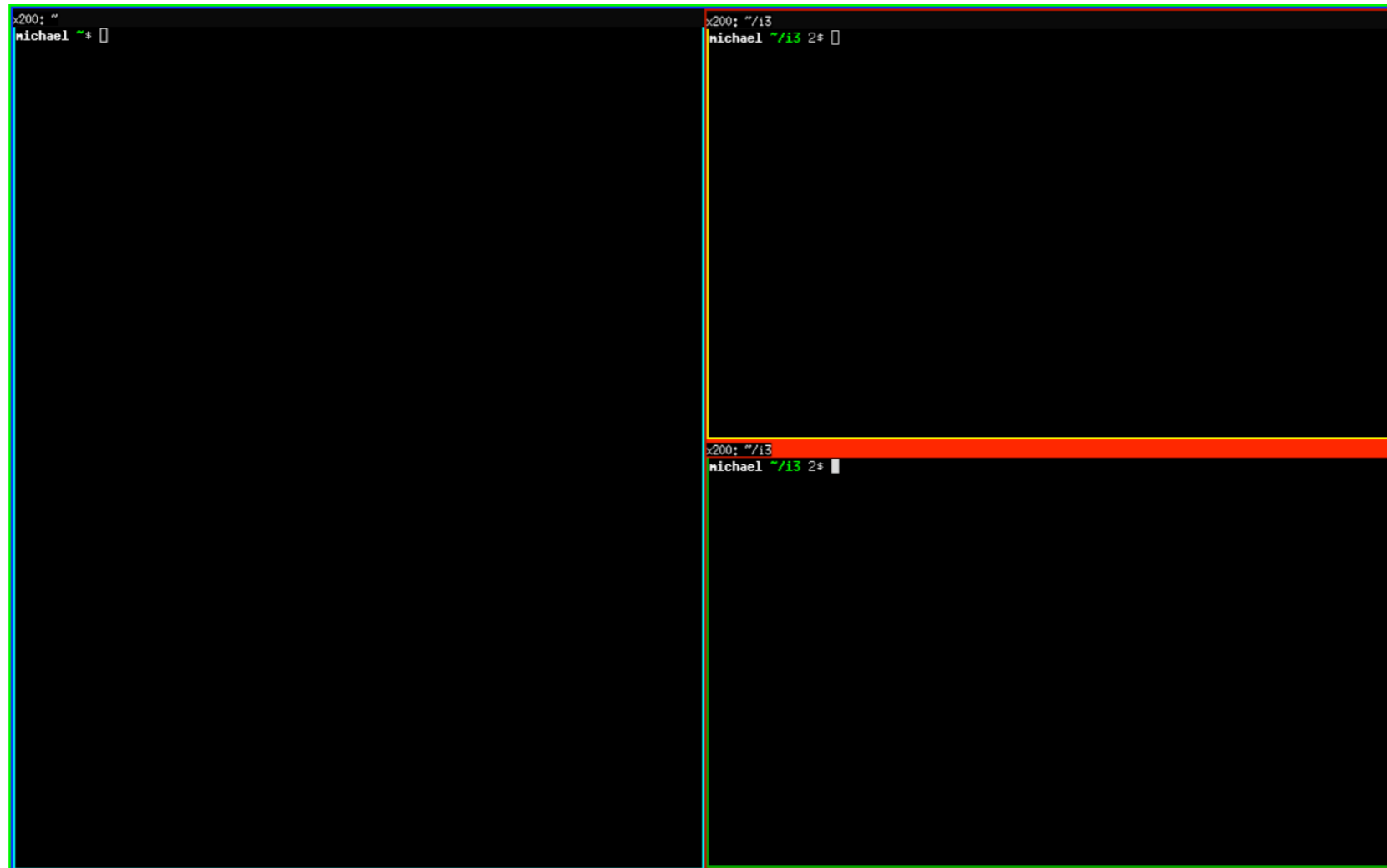
Let's open two terminals (alt+enter):

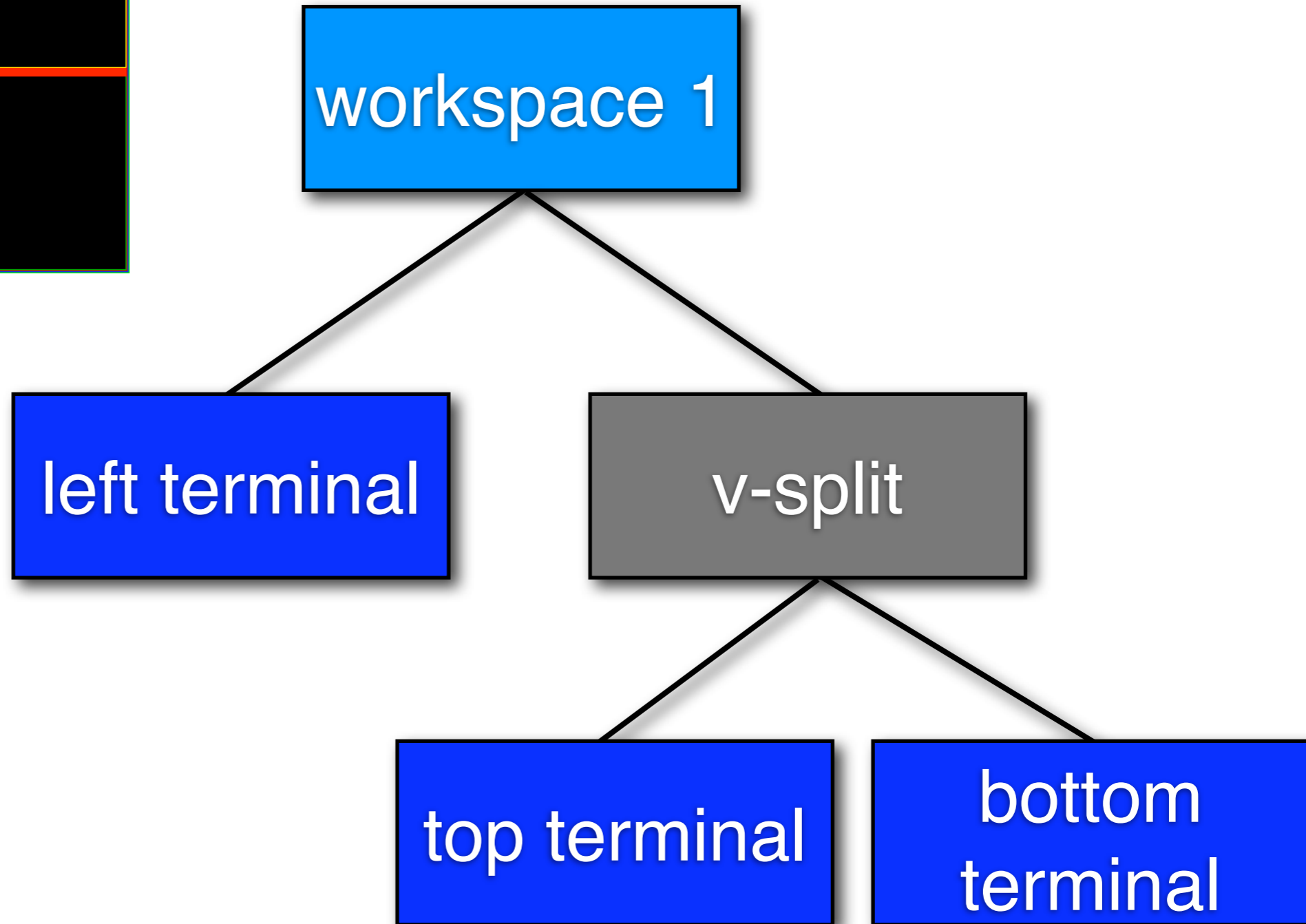
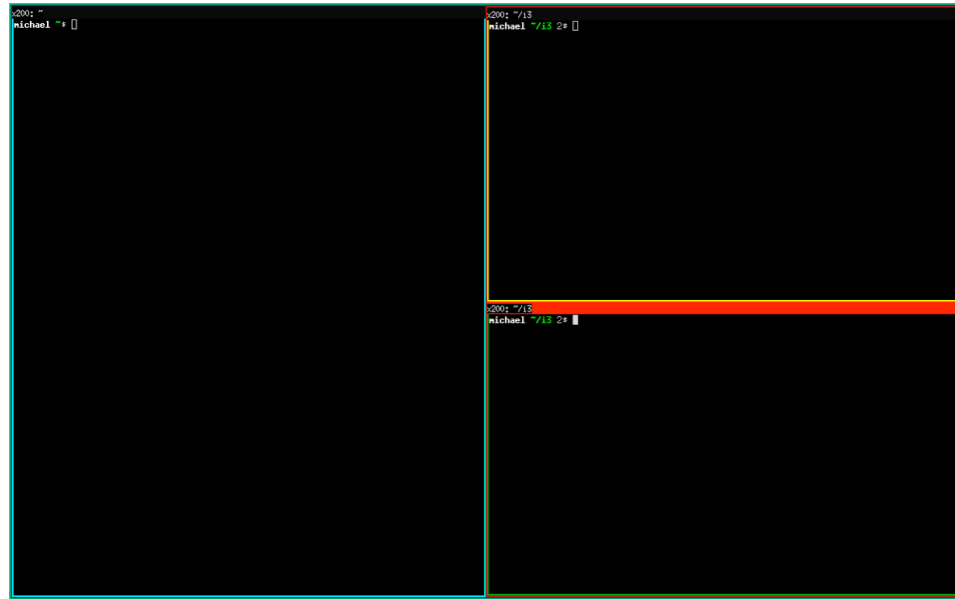


default mode is horizontal tiling
(most displays are wide-screen nowadays)

NEW: Every window you see is in a container.
This means you can split everything.

Let's split the current container vertically (alt+v)
and open a new terminal (alt+enter):





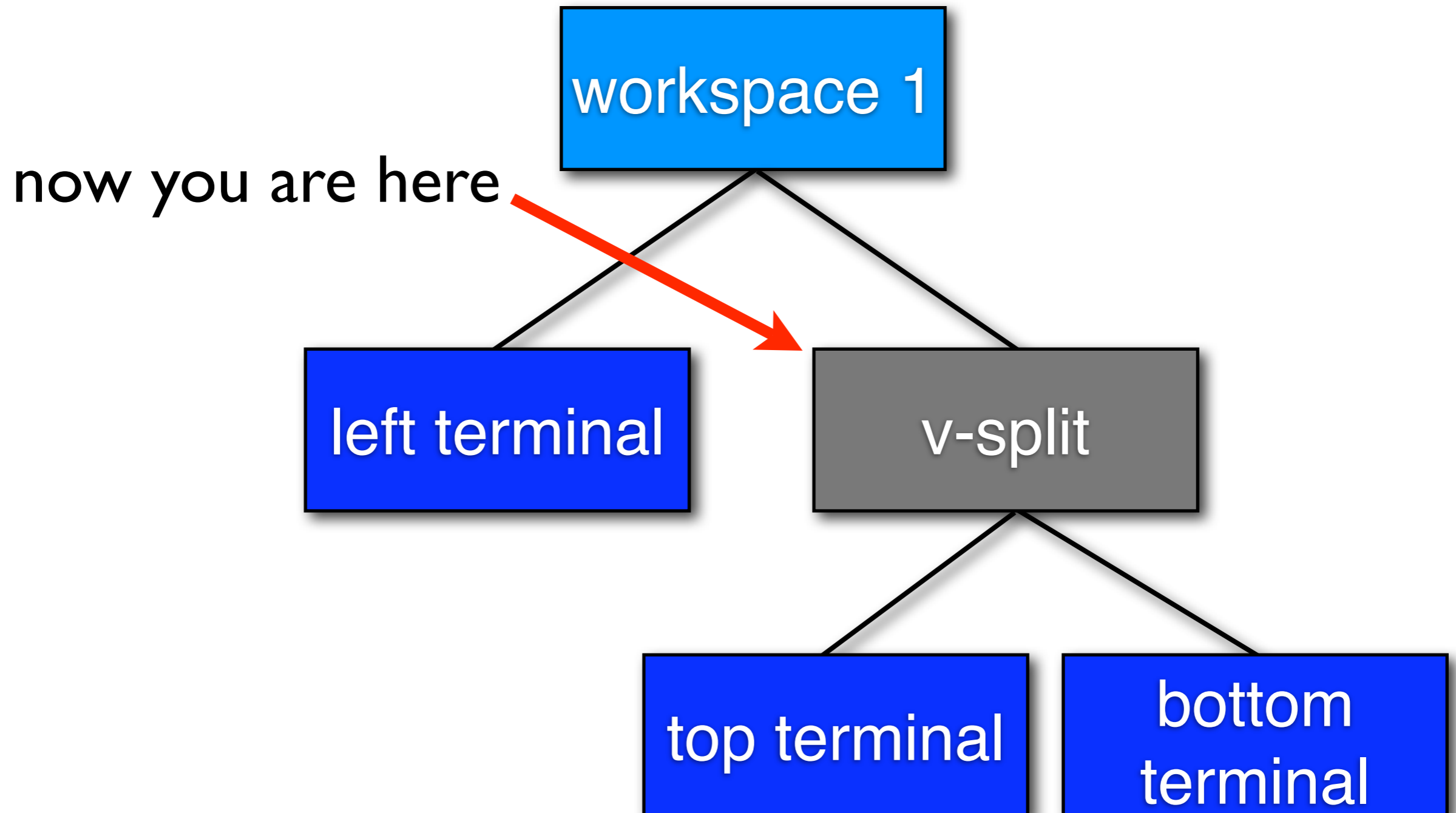
in the tree, you are here now

But wait a second?

**The new window automatically opened
in the split container.**

**So, how can we open a new window next
to the split container?**

Just use level up (alt+u)!



...then open a new terminal (alt+enter):

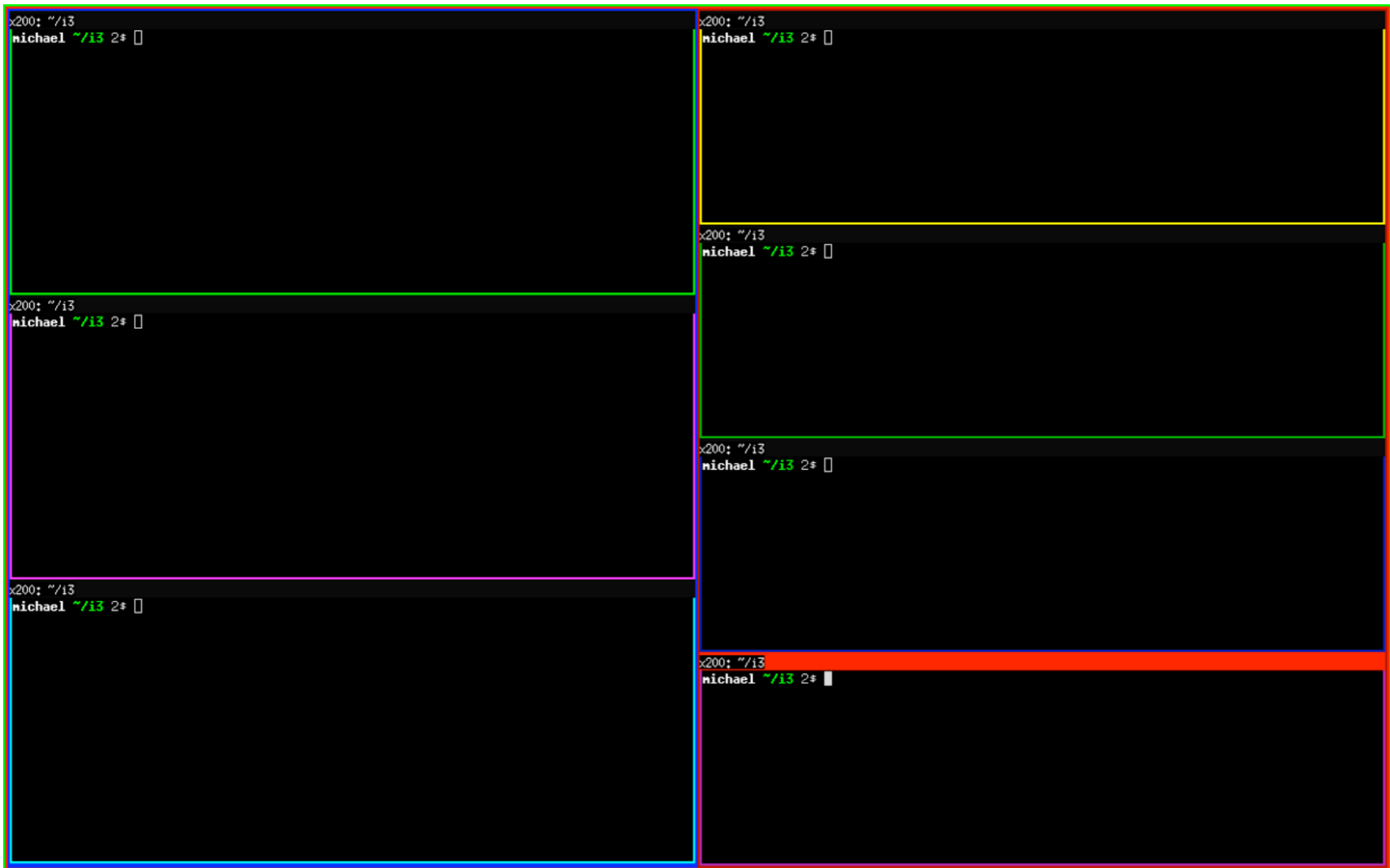


3: some layouts

The tree branch is supposed to make things easier. Let's have a look at some common layouts and how to create them.

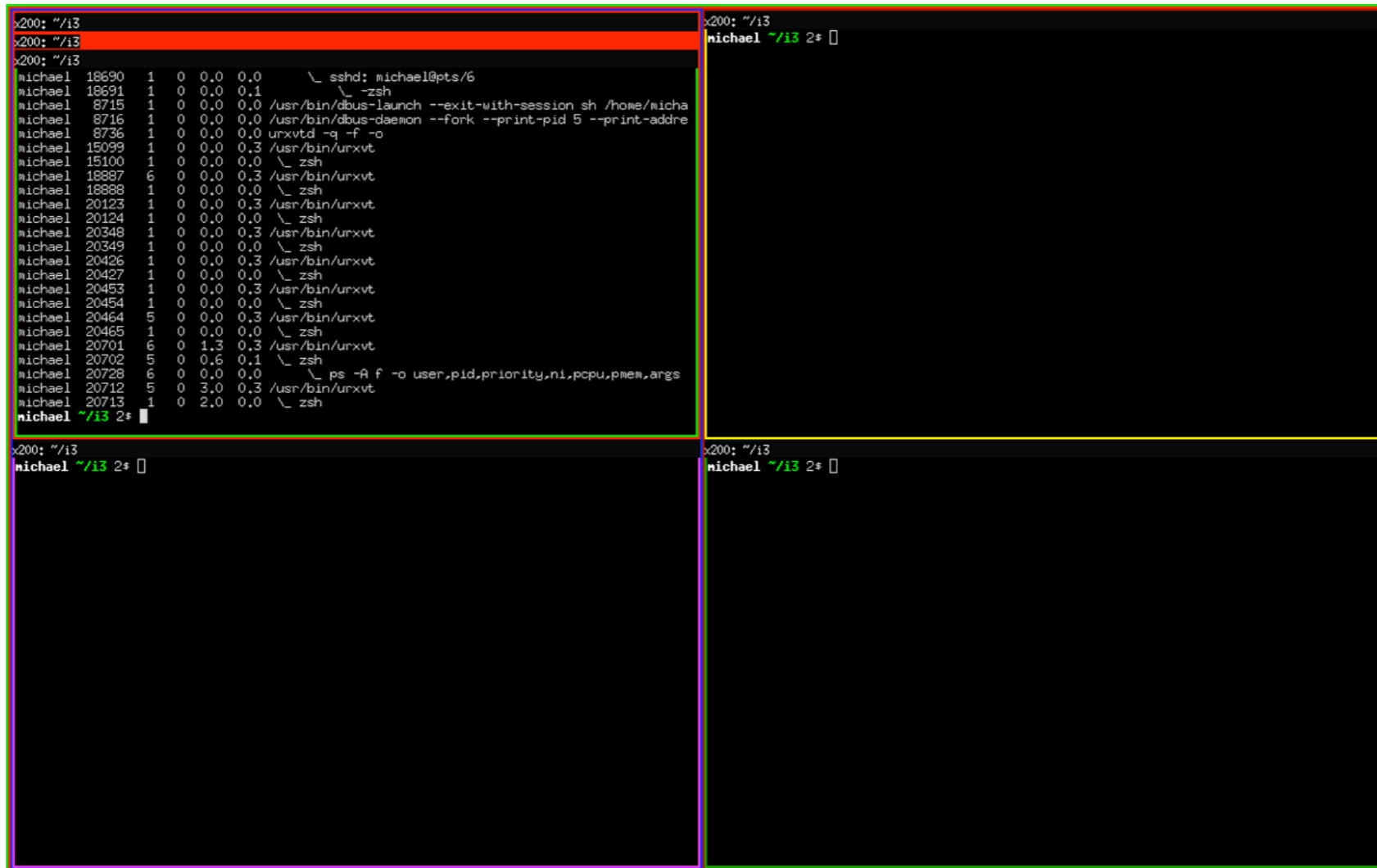
(of course, there is more than one way to do it)

lots of terminals, aligned in two columns:



howto: open, split v, open, open, (left col done)
level up, open, split v, open, open

grid layout with multiple stacks inside



The image displays a 2x2 grid of terminal windows. The top-left window shows a process list with columns for user, PID, PPID, priority, nice, and command. The top-right window shows a shell prompt. The bottom-left and bottom-right windows also show shell prompts.

```
x200: ~/i3
x200: ~/i3
x200: ~/i3
michael 18690 1 0 0.0 0.0 \ sshd: michael@pts/6
michael 18691 1 0 0.0 0.1 \ zsh
michael 8715 1 0 0.0 0.0 /usr/bin/dbus-launch --exit-with-session sh /home/micha
michael 8716 1 0 0.0 0.0 /usr/bin/dbus-daemon --fork --print-pid 5 --print-addre
michael 8736 1 0 0.0 0.0 urxvtd -q -f -o
michael 15099 1 0 0.0 0.3 /usr/bin/urxvt
michael 15100 1 0 0.0 0.0 \ zsh
michael 18887 6 0 0.0 0.3 /usr/bin/urxvt
michael 18888 1 0 0.0 0.0 \ zsh
michael 20123 1 0 0.0 0.3 /usr/bin/urxvt
michael 20124 1 0 0.0 0.0 \ zsh
michael 20348 1 0 0.0 0.3 /usr/bin/urxvt
michael 20349 1 0 0.0 0.0 \ zsh
michael 20426 1 0 0.0 0.3 /usr/bin/urxvt
michael 20427 1 0 0.0 0.0 \ zsh
michael 20453 1 0 0.0 0.3 /usr/bin/urxvt
michael 20454 1 0 0.0 0.0 \ zsh
michael 20464 5 0 0.0 0.3 /usr/bin/urxvt
michael 20465 1 0 0.0 0.0 \ zsh
michael 20701 6 0 1.3 0.3 /usr/bin/urxvt
michael 20702 5 0 0.6 0.1 \ zsh
michael 20728 6 0 0.0 0.0 \ ps -A f -o user,pid,priority,nl,pcpu,pmem,args
michael 20712 5 0 3.0 0.3 /usr/bin/urxvt
michael 20713 1 0 2.0 0.0 \ zsh
michael ~/i3 2#
```

```
x200: ~/i3
michael ~/i3 2#
```

```
x200: ~/i3
michael ~/i3 2#
```

```
x200: ~/i3
michael ~/i3 2#
```

howto: open, split v, open, (left col done)
level up, open, split v, open (right col done)
split v, open, open, stacking (for each cell)

one big window on the bottom, two above



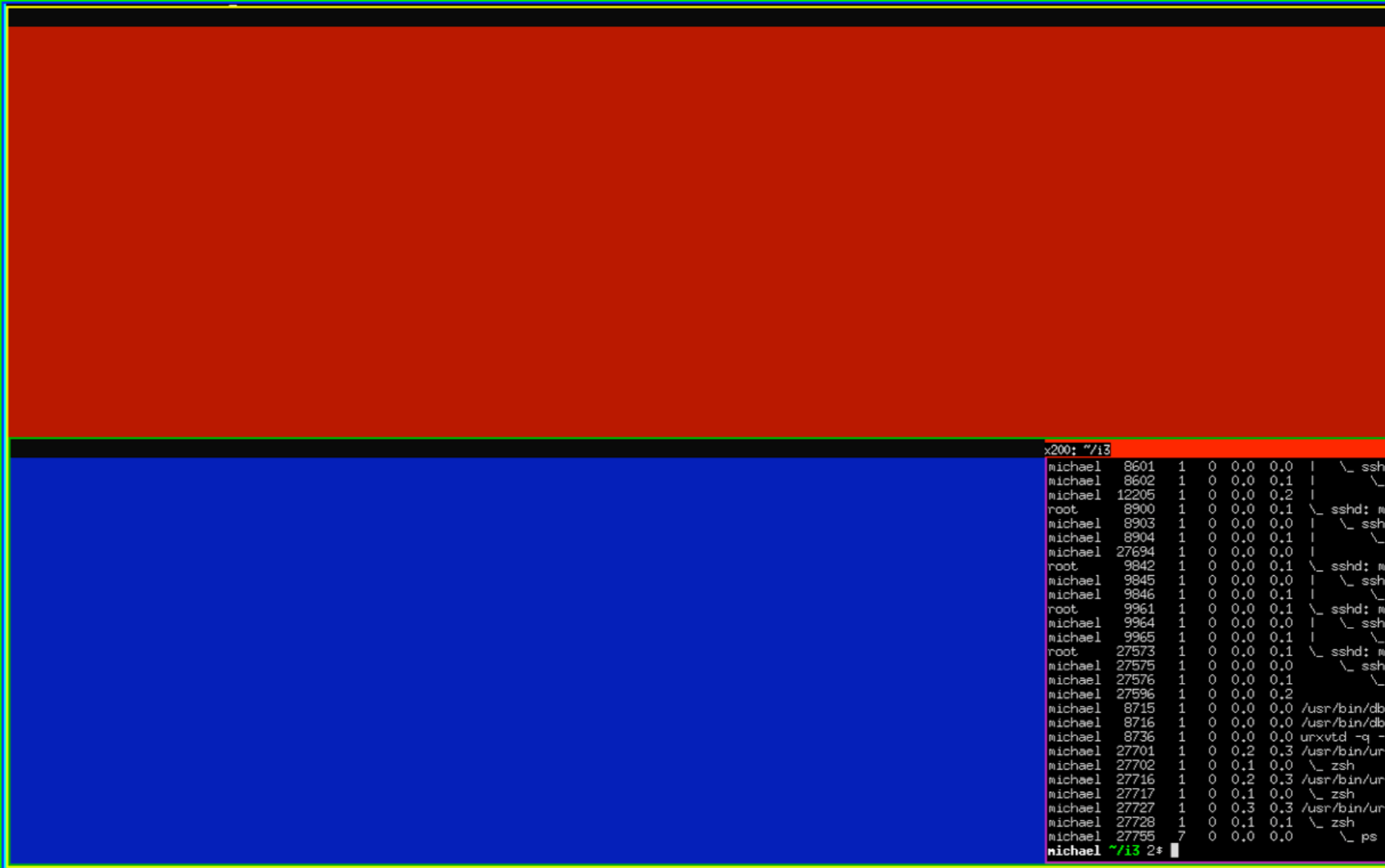
howto: split v, open, open (two rows done),
go up, split h, open (top terminals done)

4: goodies

If it was only for the layout, we probably would not have done the refactoring.

However, a lot of cool stuff is possible now.

Resize everything



A terminal window with a red background for the top half and a blue background for the bottom half. The bottom half displays a process list with columns for user, PID, PPID, CPU, MEM, and command. The list includes processes for 'ssh', 'sshd', 'zsh', and 'ps'.

```
x200: ~/i3
michael 8601 1 0 0.0 0.0 | \ ssh
michael 8602 1 0 0.0 0.1 | \
michael 12205 1 0 0.0 0.2 | \
root 8900 1 0 0.0 0.1 | \ sshd: m
michael 8903 1 0 0.0 0.0 | \ ssh
michael 8904 1 0 0.0 0.1 | \
michael 27694 1 0 0.0 0.0 | \
root 9842 1 0 0.0 0.1 | \ sshd: m
michael 9845 1 0 0.0 0.0 | \ ssh
michael 9846 1 0 0.0 0.1 | \
root 9961 1 0 0.0 0.1 | \ sshd: m
michael 9964 1 0 0.0 0.0 | \ ssh
michael 9965 1 0 0.0 0.1 | \
root 27573 1 0 0.0 0.1 | \ sshd: m
michael 27575 1 0 0.0 0.0 | \ ssh
michael 27576 1 0 0.0 0.1 | \
michael 27596 1 0 0.0 0.2 | \
michael 8715 1 0 0.0 0.0 /usr/bin/db
michael 8716 1 0 0.0 0.0 /usr/bin/db
michael 8736 1 0 0.0 0.0 urxvtc -q -
michael 27701 1 0 0.2 0.3 /usr/bin/ur
michael 27702 1 0 0.1 0.0 \ zsh
michael 27716 1 0 0.2 0.3 /usr/bin/ur
michael 27717 1 0 0.1 0.0 \ zsh
michael 27727 1 0 0.3 0.3 /usr/bin/ur
michael 27728 1 0 0.1 0.1 \ zsh
michael 27755 7 0 0.0 0.0 \ ps
michael ~/i3 2#
```

NEW: You can now resize everything.

Fullscreen everything



```
x200: ~/i3
x200: ~/i3
michael ~/i3 2# figlet 'full screen stack'
Full Screen Stack
michael ~/i3 2#
```

NEW: You can put everything into fullscreen mode.

Restore layout parts

In addition to providing the tree as JSON, i3 can also read it back from JSON.

This way, you can restore a layout at any place in the tree (container, workspace, ...).

This is handy for applications which have many windows, like GIMP for example.

Survive inplace restarts

Thanks to the layout restore feature, surviving an inplace restart without losing your layout is easy: i3 stores the layout in `~/.i3/` before restarting and restores it afterwards.

Hooray!

Session saving

With the layout restore function and some scripting magic, it should be manageable to implement a session saver.

New command parser

The old cryptic commands are replaced by explicit commands:

[next|prev] [h|v]

move [before|after] [h|v]

level [up|down]

layout [default|stacking|tabbed]

workspace <name>

fullscreen

kill

restore <path-to-layout>

...

New command parser

Also, you can now chain commands:

```
workspace 3; exec /usr/bin/xterm -e cmus
```

And make them match only specific windows:

```
[title="*Firefox*"] kill
```