

Configuring a Pretty and Usable Terminal Emulator for WSL

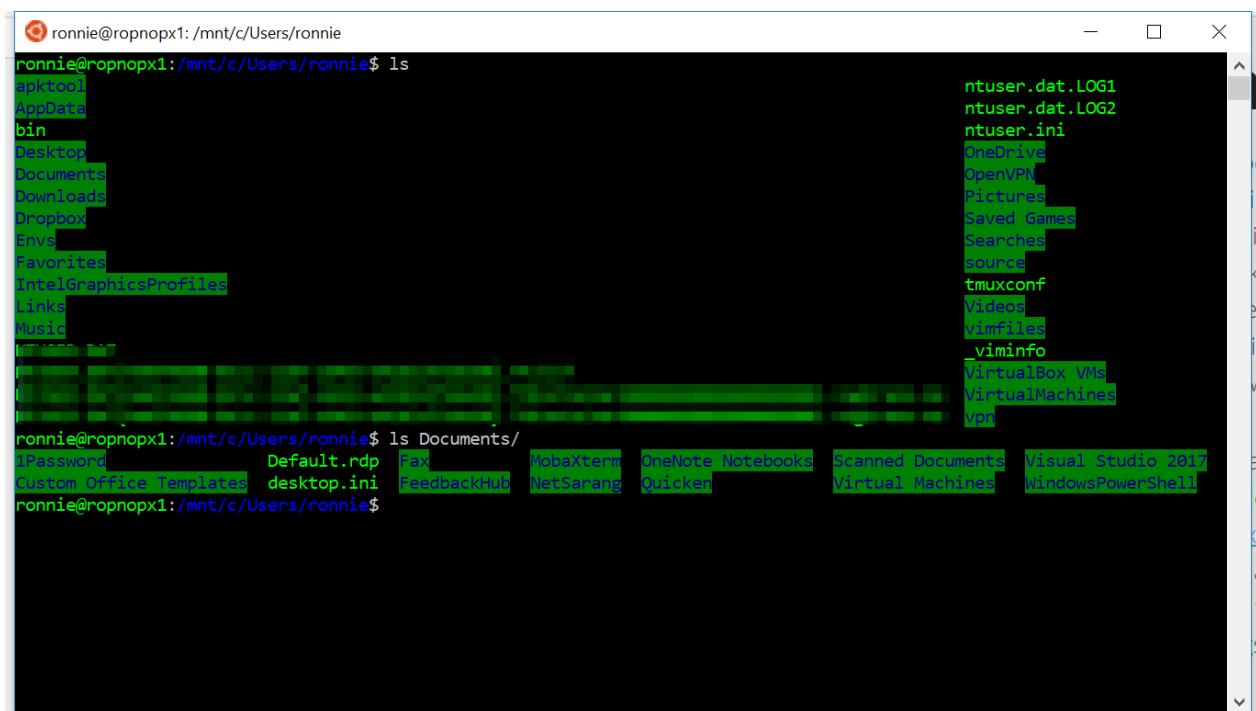
 blog.ropnop.com/configuring-a-pretty-and-usable-terminal-emulator-for-wsl

Introduction

I've been using a Mac as my daily driver for work for the last few years. While there's nothing particularly special about MacOS that I love (in fact there's quite a bit I *don't* like), it's honestly been the terminal and the underlying Unix based operating system that keep me glued to it. With Homebrew, command line tools *just work*. Python and Node dev environments *just work*. And using iTerm2 with oh-my-zsh is the best terminal experience I've ever had. I often feel like I just pay the premium for Mac hardware to have a reliable and easy to configure *Nix operating system.

But lately I've really been wanting to get off the Mac ecosystem and start using Windows 10 on my X1 Carbon as my daily machine. With the Windows Subsystem for Linux (WSL) it's now possible to have a "native" Ubuntu command line on my Windows 10 machine to use for my CLI nerdiness. But the only thing holding me back was the lack of a nice terminal emulator (admittedly, I'm shallow and like pretty things).

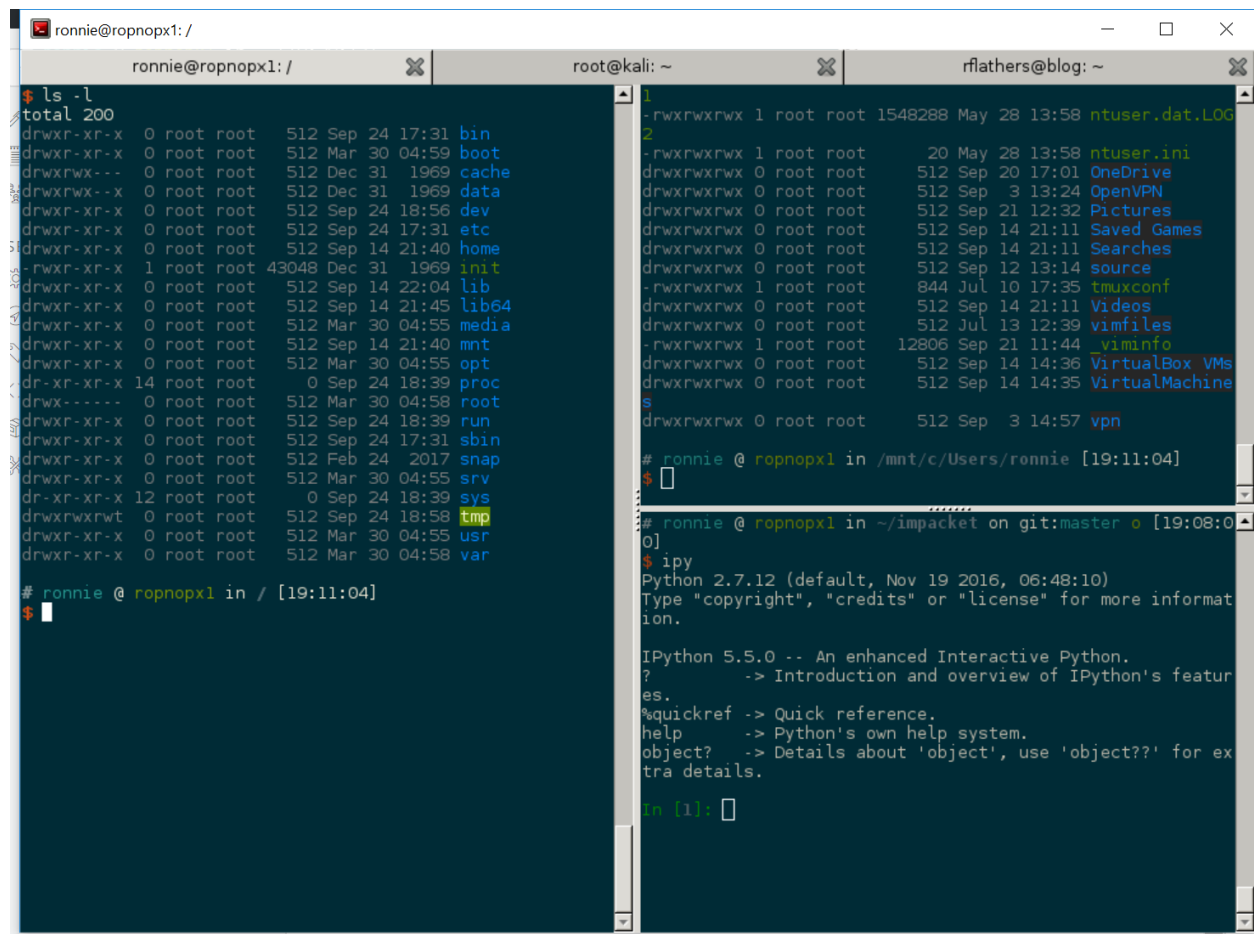
This just wasn't going to cut it:



```
ronnie@ropnopx1: /mnt/c/Users/ronnie$ ls
apktool                               ntuser.dat.LOG1
AppData                               ntuser.dat.LOG2
bin                                   ntuser.ini
Desktop                              OneDrive
Documents                            OpenVPN
Downloads                             Pictures
Dropbox                              Saved Games
Envvars                              Searches
Favorites                             source
IntelGraphicsProfiles                tmuxconf
Links                                 Videos
Music                                 vimfiles
ntuser.dat                           viminfo
ntuser.dat.LOG1                     VirtualBox VMs
ntuser.dat.LOG2                     VirtualMachines
ntuser.ini                           vpn
OneDrive                             Virtual Machines
OpenVPN                             Visual Studio 2019
Pictures                             WindowsPowerShell
Saved Games
Searches
source
tmuxconf
Videos
vimfiles
viminfo
VirtualBox VMs
VirtualMachines
vpn

ronnie@ropnopx1:/mnt/c/Users/ronnie$ ls Documents/
IPassword                        Default.rdp      Fax              MobaXterm       OneNote Notebooks  Scanned Documents  Visual Studio 2019
Custom Office Templates          desktop.ini      FeedbackHub      NetSarang        Quicker            Virtual Machines   WindowsPowerShell
ronnie@ropnopx1:/mnt/c/Users/ronnie$
```

After much tinkering, I've ended up with what I feel is the most comfortable terminal experience I can get on Windows. It supports tabs, splits, mouse mode and has a pretty color scheme to boot:



In this post, I'm going to quickly explain how I got it running and configured, and some of the other options I tried.

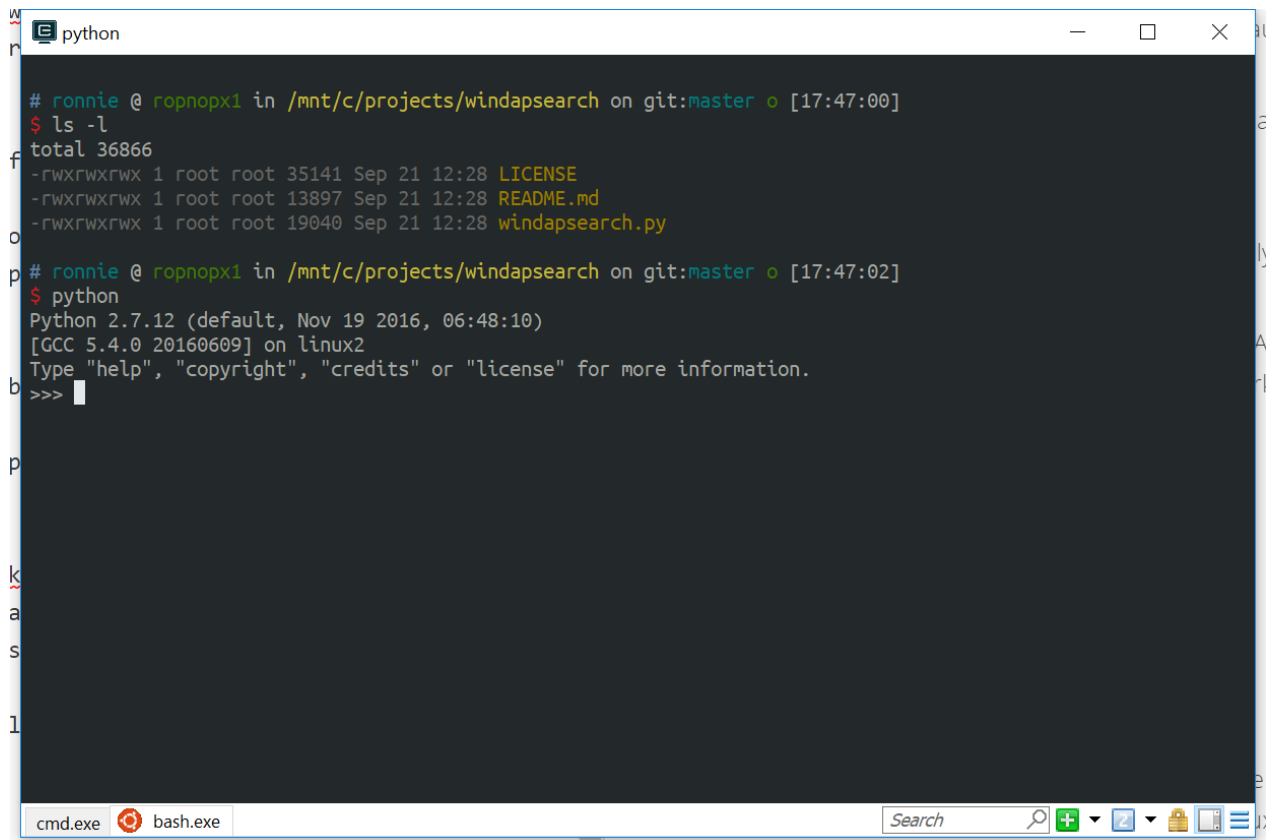
First Attempts

I really just wanted the equivalent of iTerm2 in Windows. I wanted to utilize WSL (not Cygwin) and at a minimum needed:

- Pretty colors and fonts
- Tabs (non-negotiable)
- Working mouse support for scrolling and Vim/Tmux
- Tmux support and auto resizing
- Sane copy/paste

I think I tried every major Windows terminal app I could find. Each had their own drawbacks and I eventually gave up. Some of the ones I tried:

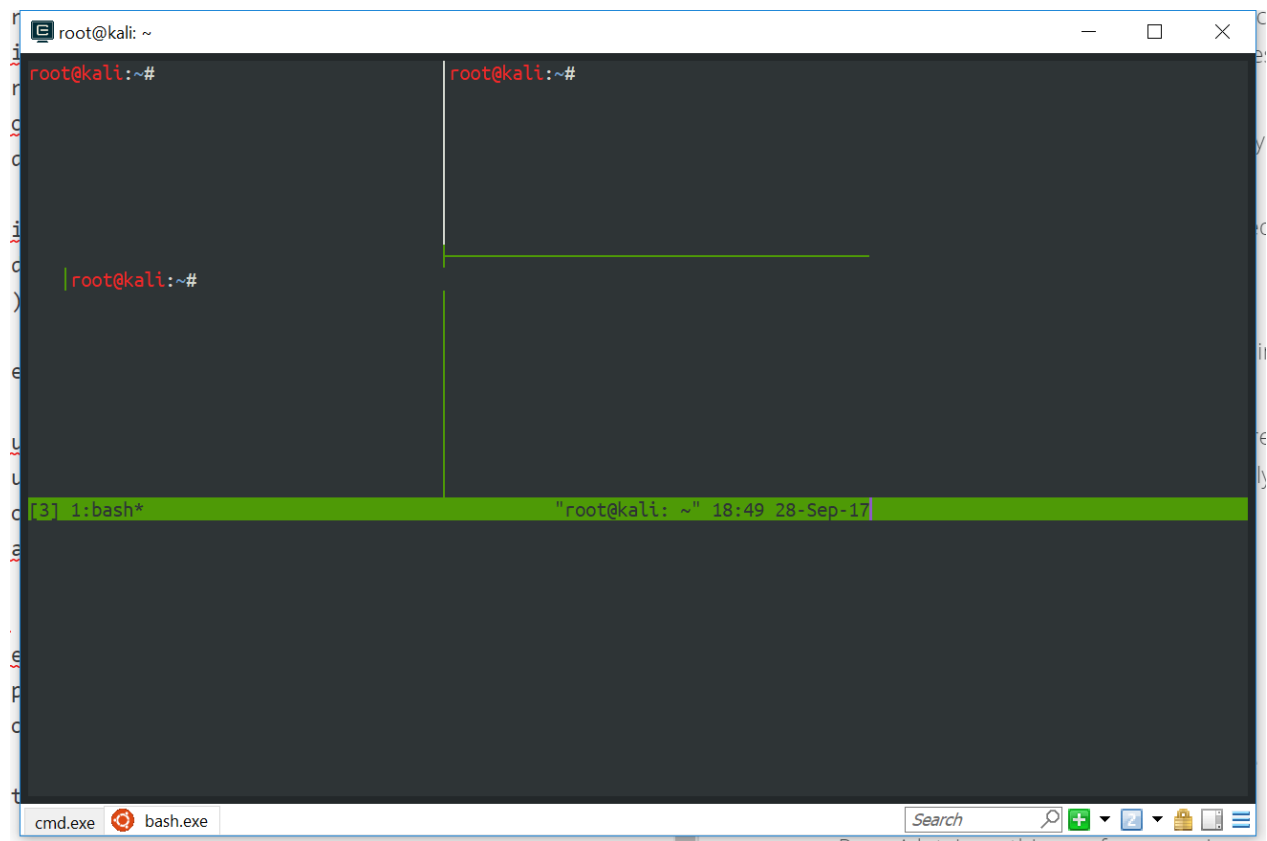
The closest I got, and one that I used for a while was Cmdr:



```
# ronnie @ ropnopx1 in /mnt/c/projects/windapsearch on git:master o [17:47:00]
$ ls -l
total 36866
-rwxrwxrwx 1 root root 35141 Sep 21 12:28 LICENSE
-rwxrwxrwx 1 root root 13897 Sep 21 12:28 README.md
-rwxrwxrwx 1 root root 19040 Sep 21 12:28 windapsearch.py

# ronnie @ ropnopx1 in /mnt/c/projects/windapsearch on git:master o [17:47:02]
$ python
Python 2.7.12 (default, Nov 19 2016, 06:48:10)
[GCC 5.4.0 20160609] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Unfortunately, when I started using Tmux it became a problem. I could never get mouse mode to work (scrolling or selecting panes), and resizing windows was problematic. I'd end up with screens like this a lot:



```
root@kali: ~#
root@kali:~#
root@kali:~#

[3] 1:bash* "root@kali: ~" 18:49 28-Sep-17
```

Not gonna cut it for me (though I still do use Cmder regularly for when I need to run Windows cmd.exe)

Linux Terminal Emulators

What I realized in my search and multiple trials was there just *wasn't* a good Windows terminal emulator. When I was about to give up, I saw a post on Reddit about someone who got XFCE working on WSL Bash. That was way overkill for what I wanted to accomplish, but reading through the post I learned/realized that if I had an X Server running on Windows, I could use GUI Linux terminal emulators “natively” on Windows! That opened up a ton of possibilities, and one of my favorite Linux terminals, Terminator, was now a possibility!

Installing an X Server

To run an X Window application, I needed to have an X Server installed and running on my Windows 10 machine. After researching, it seemed the two most popular options are:

- VcXsrv
- Xming

I went with VcXsrv since it looked like it was more actively maintained, but I tried both and they work the same.

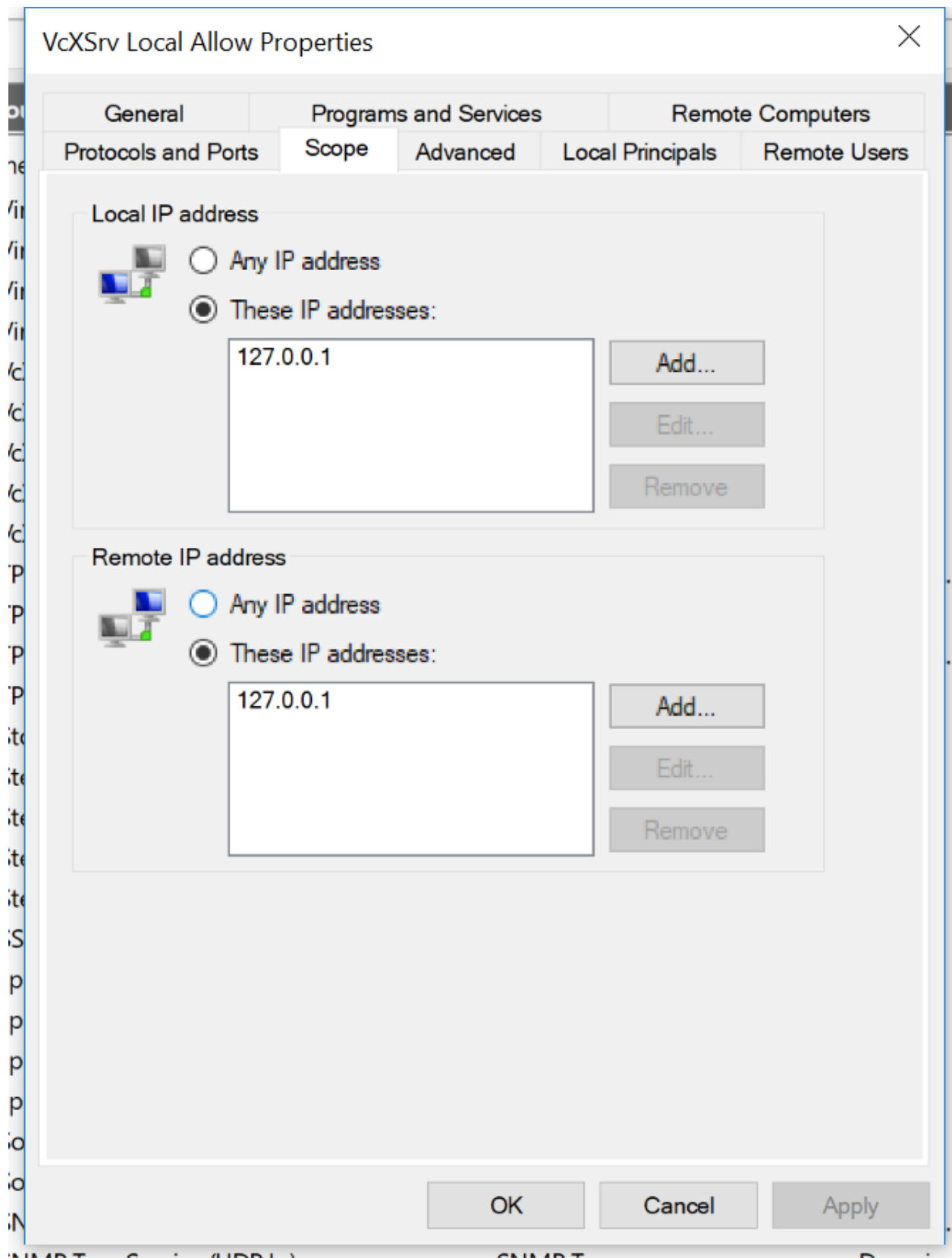
After installing, VcXsrv creates a desktop shortcut to start the server in multi-window mode through the following command:

```
"C:\Program Files\VcXsrv\vcxsrv.exe" :0 -ac -terminate -lesspointer -  
multiwindow -clipboard -wgl -dpi auto
```

A taskbar icon shows it's running, and we can verify by looking at netstat:

```
C:\Windows\system32>netstat -abno|findstr 6000  
TCP      0.0.0.0:6000      0.0.0.0:0        LISTENING  
6216
```

It's important to note that VcXsrv is listening on all interfaces and requests a blanket firewall exception for private networks. AFAIK there is no way to only force it to listen/accept connections from localhost only, so I disallowed the firewall exception request and configured a custom rule to only allow traffic from **127.0.0.1**



Configuring Terminator

Once VcXsrv was installed and configured to allow access from `127.0.0.1`, the next step was to install Terminator on WSL Bash:

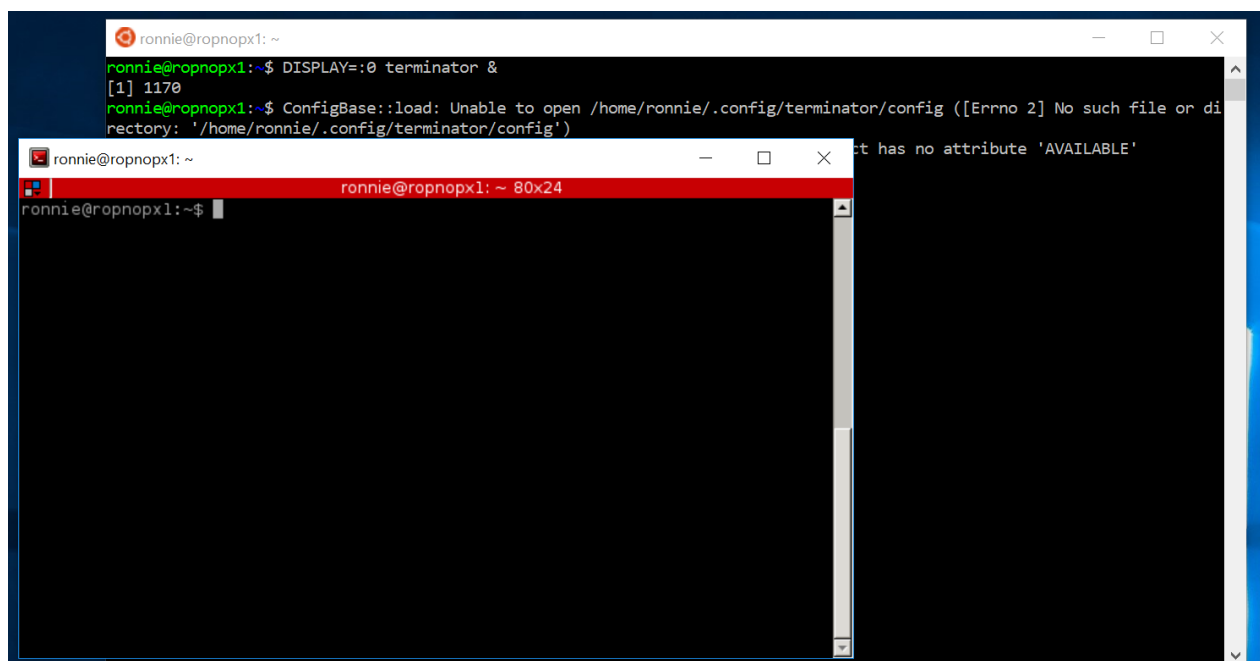
```
$ sudo apt-get install  
terminator
```

If I didn't want to use terminator, any other terminal emulator should work, including Gnome Terminal (which Terminator is based on), Urxvt, or xterm.

After it installed, all that was left was to try launching it by specifying the X Display to connect to (:0)

```
$ DISPLAY=:0  
terminator &
```

And a nice Terminator windows popped up :)



Installing Zsh

The next step I did was install Zsh with [oh-my-zsh](#). Installation is straightforward:

```
sudo apt-get install curl wget git zsh  
curl -L https://raw.githubusercontent.com/robbyrussell/oh-my-zsh/master/tools/install.sh | bash
```

I set the theme "ys" in `.zshrc`

The only "gotcha" about using Bash in WSL is it will always run Bash instead of Zsh. To get around that, I add this to the end of my `.bashrc` which will launch `zsh` instead when it starts up:

```
if [ -t 1 ];  
then  
    exec zsh  
fi
```

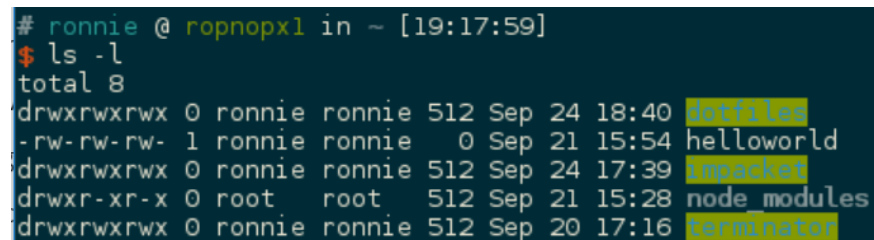
Terminator Colorscheme

The next thing I chose to do was change the default Terminator colorscheme to Solarized Dark. The easiest way to do this IMO, is to use the awesome node package [base16-builder](#)

```
sudo apt-get install nodejs-legacy  
sudo npm install --global base16-builder  
mkdir -p .config/terminator  
base16-builder -s solarized -t terminator -b dark >  
.config/terminator/config
```

Dircolors

It was looking good so far, but the dircolors were still awful:



```
# ronnie @ ropnopx1 in ~ [19:17:59]  
$ ls -l  
total 8  
drwxrwxrwx 0 ronnie ronnie 512 Sep 24 18:40 .dotfiles  
-rw-rw-rw- 1 ronnie ronnie  0 Sep 21 15:54 helloworld  
drwxrwxrwx 0 ronnie ronnie 512 Sep 24 17:39 .mpack  
drwxr-xr-x 0 root  root  512 Sep 21 15:28 node_modules  
drwxrwxrwx 0 ronnie ronnie 512 Sep 20 17:16 terminator
```

I found [Solarized dircolors](#) on Github and downloaded them to `.dir_colors`

```
wget https://raw.githubusercontent.com/seebi/dircolors-solarized/master/dircolors.256dark  
mv dircolors.256dark .dir_colors
```

Then lastly, added this to my `.zshrc` to eval the Solarized dircolors on startup:

```
if [ -f ~/.dir_colors ];  
then  
    eval `dircolors`  
    ~/.dir_colors`  
fi
```

Finally I had a pretty enough terminal to my liking. Still no iTerm2, but pretty damn close IMO

Launching Terminator Directly

The final hurdle was figuring out a way to launch Terminator directly without having to open a Bash window first and then typing `DISPLAY=:0 terminator &`

From the command line, I could launch bash with the `-c` parameter and it would work:

```
C:\Users\ronnie>bash -c -l "DISPLAY=:0  
terminator &"
```

However, this still required me to open a command window. After a lot of trial and error and research on StackOverflow, I figured out how to launch a hidden command window using the WShell Object in VBS. The final script I used is here:

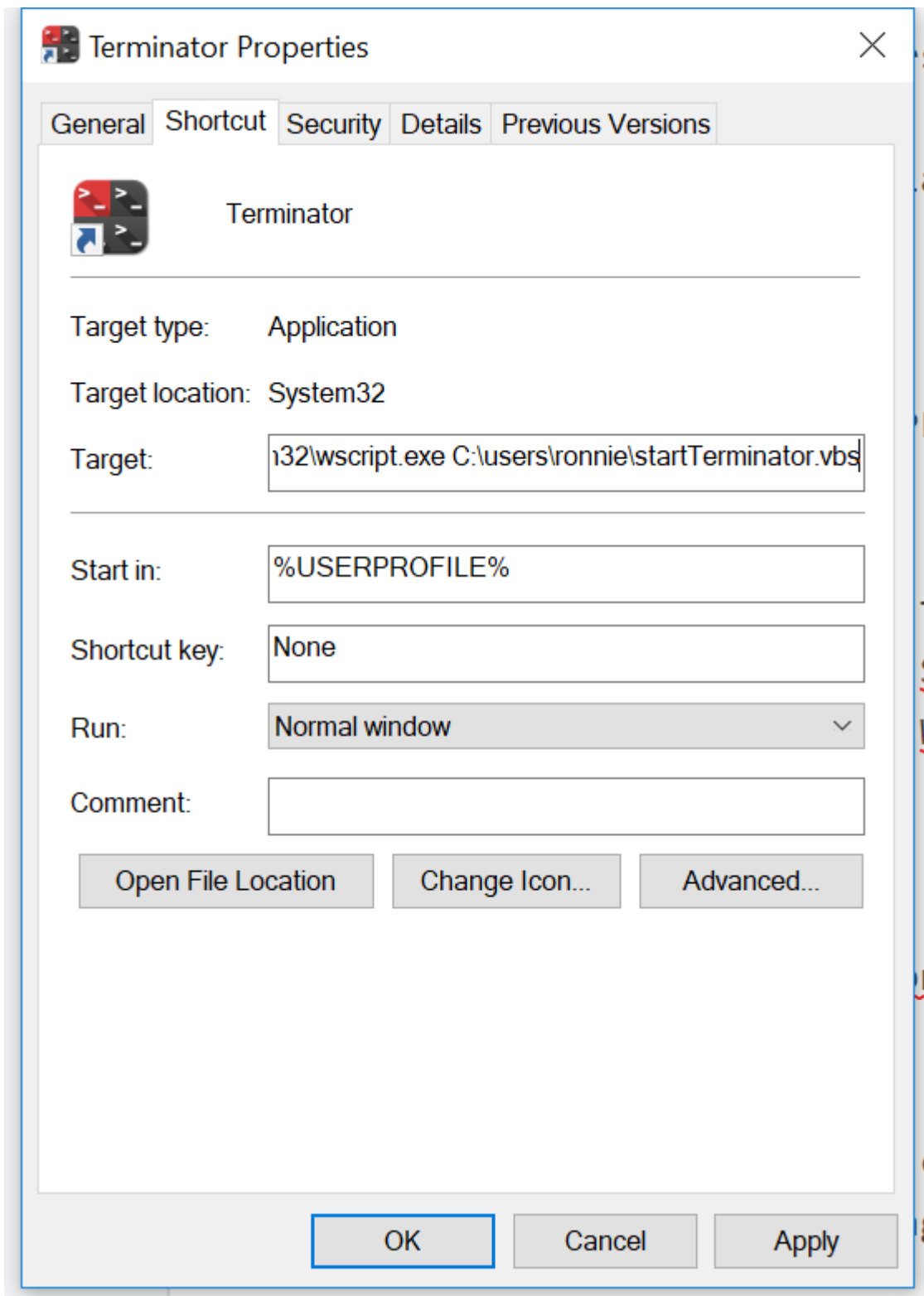
```
args = "-c" & " -l " & """"DISPLAY=:0 terminator""""
```

```
WScript.CreateObject("Shell.Application").ShellExecute "bash", args, "", "open", 0
```

[view raw startTerminator.vbs](#) hosted with ❤ by [GitHub](#)

With that script, I was able to create a Desktop shortcut to `wscript.exe` and execute it with the following command:

```
C:\Windows\System32\wscript.exe  
C:\Users\ronnie\startTerminator.vbs
```

The last step was to find a pretty Terminator icon file for the shortcut and add it to my taskbar for quick launching.

One final note is the "Start In" option. It's impossible to have Terminator start in my Linux home directory through this method since that path is not "known" to Windows. To get around it, I added this to my `.zshrc` so it CD's to my home directory on startup:

```

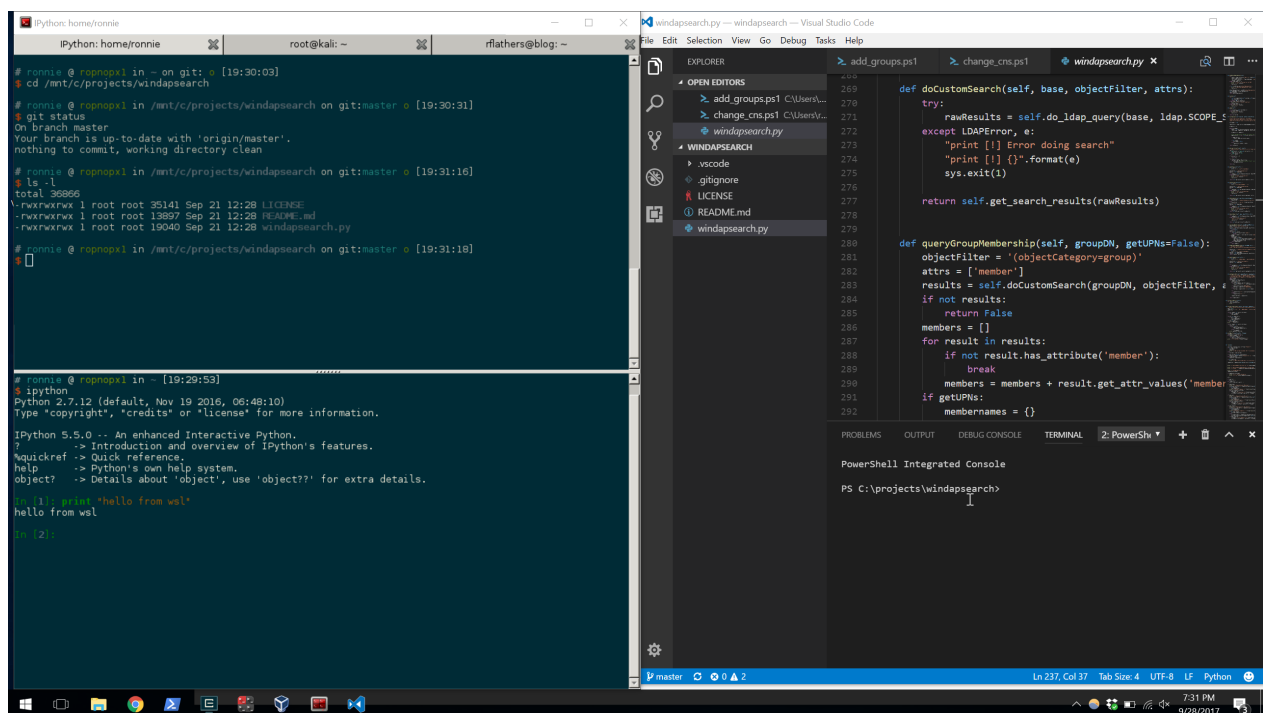
if [ -t 1 ];
then
    cd ~
fi

```

Hacky, but it works.

Conclusions

I've been using Terminator for WSL for a while now and am loving it. It's the best terminal experience I've been able to get on Windows so far. As long as I have VcXsrv running I've had no issues with launching it and it runs very smoothly. Terminator is very configurable, and I've configured and changed some of the keyboard shortcuts to suit my liking. Overall I love having the splits/panes/tabs and when I'm SSH'd into multiple boxes through WSL it's amazing. My workflow now is generally running VS Code on my Windows, with multiple Terminator panes open to `/mnt/c/projects/whatever` and being SSH'd into my lab. Love it.



Hope this helps someone! Let me know if you have a different/better solution!

-ropro

[wsl](#) [bash](#) [zsh](#) [terminator](#)

- [← Previous Post](#)
- [Next Post →](#)