Computer Setup Programming

Robin Hellmers

July 16, 2020

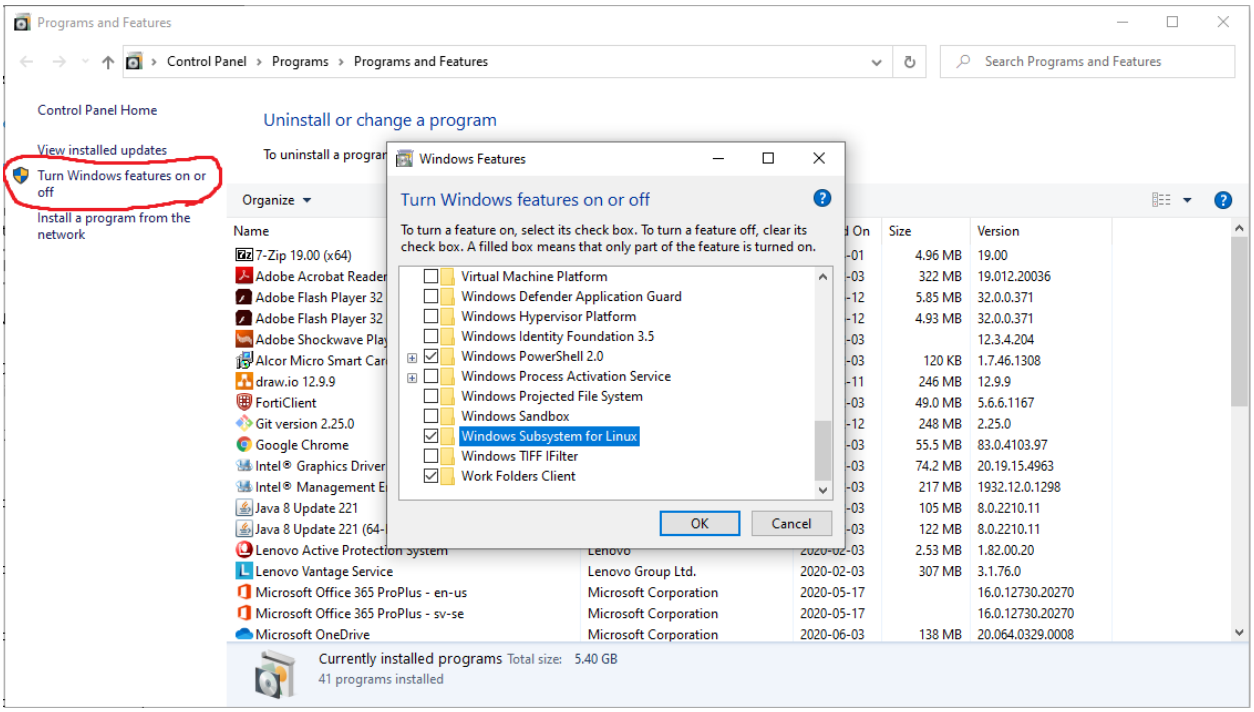
Contents

1 Ubuntu App - Windows Subsystem for Linux

1.1 Installation Ubuntu WSL with Visual Studio Code

`https://code.visualstudio.com/docs/cpp/config-wsl`

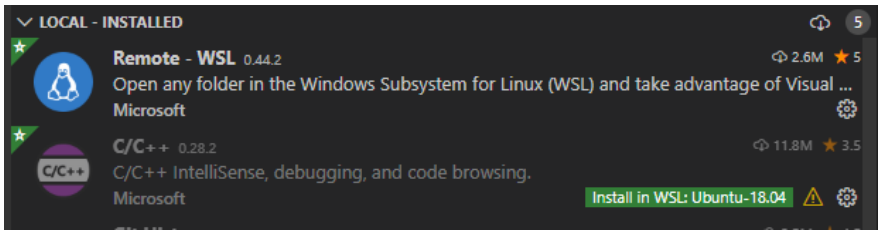
1. Download Ubuntu 18.04 LTS from Windows Store.
2. Activate `Windows Subsystem for Linux` through `Programs and Features`.



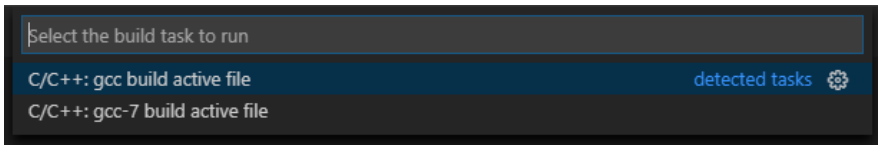
3. Restart computer.
4. Run `Ubuntu 18.04 LTS` and let it install. Might have to press `enter` after a while.
5. Create user with password.
6. Install `gcc` and `gdb` on the Windows Subsystem for Linux (WSL).
   - Run `sudo apt update`
   - Then `sudo apt install build-essential`
   - Or `sudo apt-get install build-essential gdb`
7. Create a new folder in the WSL where you create a C file `helloworld.c`. New folder is necessary for Visual Studio Code to realise that there is a C compiler to setup later on as it uses the open file to do the configurations.
8. Open up Visual Studio Code.
9. Install two extensions in VS code:
   - `C/C++` from Microsoft

- `Remote - WSL` from Microsoft

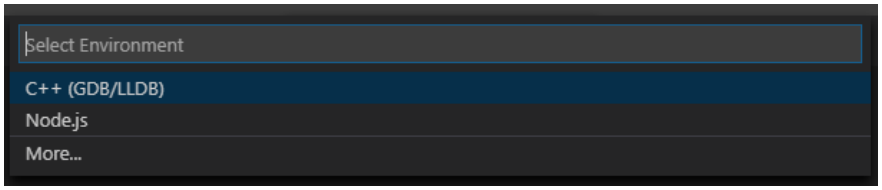
10. Press on the new icon on the left, `Remote explorer`. Right-click the `Ubuntu 18.04` and press `Connect to WSL`. A new window will appear with some connection to the WSL.
11. Press on the extension icon the left in the new window. Press `Install in WSL: Ubuntu-18.04` button on the `C/C++` extension.



12. By now it might prompt that you have to reload the window. Press that button.
13. Open up the folder you created the main C file in. `File->Open Folder...`
14. Open up the `helloworld.c` file in the file explorer.
15. Press `Terminal->Configure Default Build Task...`. In the dropdown list that should appear, choose `C/C++: gcc build active file` (Not gcc-7). A file `tasks.json` will be created, let that file be.



16. Build file with `Ctrl+Shift+b`. Press the `+` sign at the terminal to open a new terminal. Run the file `./helloworld` to test that everything is working.
17. Now onto debugging. Press `F5` or `Run->Start Debugging`. In the dropdown list that should appear, choose `C++ (GDB/LLDB)`.



18. Down at the `Output` and `Terminal`, press the three dots `...` and choose `Debug Console` in which one can run the standard `gdb` commands.

1.2 WSL Ubuntu installations

Start with:

```
sudo apt update
```

1.2.1 Compiler gcc & g++

Compiler gcc and g++ installation:

```
sudo apt install build-essential
```

1.2.2 Git

Git installation:

```
sudo apt install git
```

```
git init
```

```
git remote add origin <remote-address>
```

Save credentials:

```
git config credential.helper store
```

Get master from remote origin:

```
git pull origin master
```

In order to not have to specify `<remote>` and `<branch>` in `git pull <remote> <branch>`, but still have to do previous pull first:

```
git branch --set-upstream-to=origin/master master
```

```
git pull
```

1.2.3 Terminal bookmark directories

Install `Apparix` (Doc. `https://www.micans.org/apparix/`) with
`sudo apt-get install apparix`

Then write `apparix --shell-examples` and copy everything except the aliases at the bottom. Paste this in `/.bashrc`

Restart console.

Bookmark current directory with `bm bookmarkname` and go to the same location with `to bookmarkname`

1.3 WSL Ubuntu Customization

1.3.1 Terminal shorten name & path, add git indication

Only this link is needed. The rest below this is the same, Github is used to easily copy the code. Github with `.bashrc` code and `git-completion.bash` forked from official Git source code:
`https://github.com/robinhellmers/computer_setup`

Use `sudo chmod +x ~/git-completion.bash` in order to give permission to the user to run it. Thereby it can run it through `./bashrc`

Instructions for git fetched from here:
`https://git-scm.com/book/id/v2/Appendix-A%3A-Git-in-Other-Environments-Git-in-Bash`

Git source code `git-completion.bash`. Copy the content of the file from official git and add it to your home folder as `git-completion.bash`:
`https://github.com/git/git/blob/master/contrib/completion/git-completion.bash`

Add this above the code that is going to be replaced:

```
export PROMPT_DIRTRIM=3
PS1_custom='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u\[\033[00m\]:
\[\033[01;34m\]\w\[\033[00m\]\$ '
```

Replace the similar code with this:

```
if [ "$color_prompt" = yes ]; then
    PS1=$PS1_custom
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt
```

Add this below the code that is going to be replaced:

```
export GIT_PS1_SHOWCOLORHINTS=true
export GIT_PS1_SHOWDIRTYSTATE=true
export GIT_PS1_SHOWUNTRACKEDFILES=true
export GIT_PS1_SHOWUPSTREAM="auto"
# PROMPT_COMMAND='__git_ps1 "\u@\h:\w" "\\\£ "'
# use existing PS1 settings
PROMPT_COMMAND=$(sed -r 's|^(.+)(\\\$\s*)$|__git_ps1 "\1" "\2"|' <<< $PS1)
```

Here is all of the above:

```
export PROMPT_DIRTRIM=3
PS1_custom='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u\[\033[00m\]:\
[\033[01;34m\]\w\[\033[00m\]\$ '


if [ "$color_prompt" = yes ]; then
    PS1=$PS1_custom
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
unset color_prompt force_color_prompt


export GIT_PS1_SHOWCOLORHINTS=true
export GIT_PS1_SHOWDIRTYSTATE=true
export GIT_PS1_SHOWUNTRACKEDFILES=true
export GIT_PS1_SHOWUPSTREAM="auto"
# PROMPT_COMMAND='__git_ps1 "\u@\h:\w" "\\\£ "'
# use existing PS1 settings
PROMPT_COMMAND=$(sed -r 's|^(.+)(\\\$\s*)$|__git_ps1 "\1" "\2"|' <<< $PS1)
```

1.3.2 Fix Ubuntu terminal colors - ColorTool

Colors in the Ubuntu App can be bad. Microsoft have released `ColorTool` to fix this. See the link below:

`https://github.com/microsoft/terminal/tree/master/src/tools/ColorTool`

Further down, `README.md` should have a title **Installing** with a link to the latest `ColorTool` release, with a built `.exe` file and a `schemes` directory. Here is the current one upon writing this:

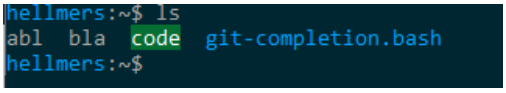
`https://github.com/microsoft/terminal/releases/tag/1904.29002`

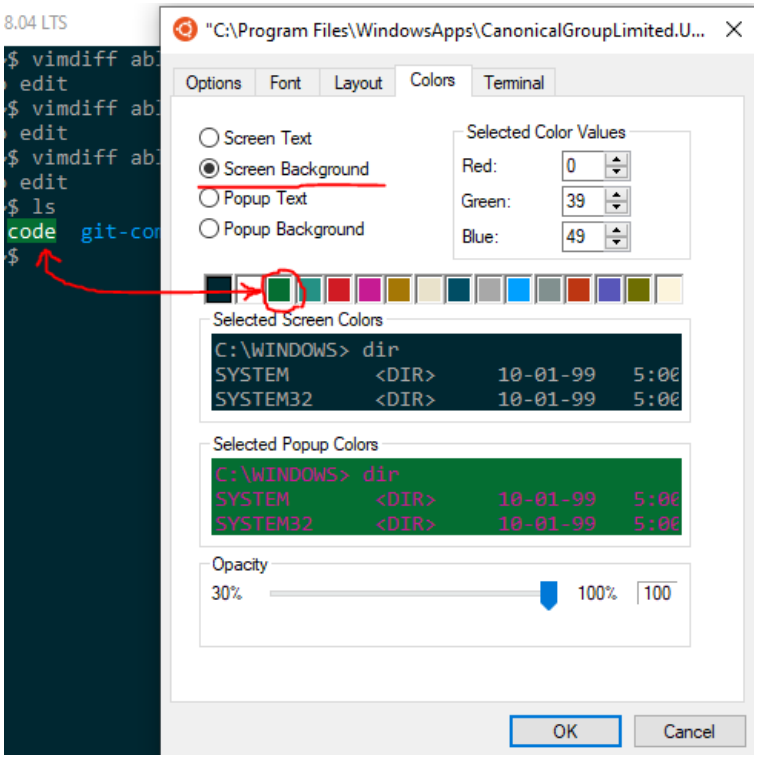
I currently use the `solarized_dark.itermcolors` scheme with some additional manual adjustment in properties, described in section 1.3.3.

1. Download the `zip` file.
2. Open `Command Prompt` in Windows.
3. Write `PATH` or `echo %PATH%` if that doesn't work, to see the different paths.
4. Find a suitable location, such as `C:\Users\Robin.Hellmers\AppData\Local\Microsoft\WindowsApps`, to unzip the files. Unzip them.
5. Open `Command prompt` at the very same directory.
6. Check the `schemes` directory for the names of the different schemes e.g. `campbell.ini`, `OneHalfDark.itermcolors`, ...
7. In the `cmd`, run `ColorTool -b {scheme}` e.g. `ColorTool.exe -b solarized_dark.itermcolors`.
8. Restart the Ubuntu app and the color scheme is applied.

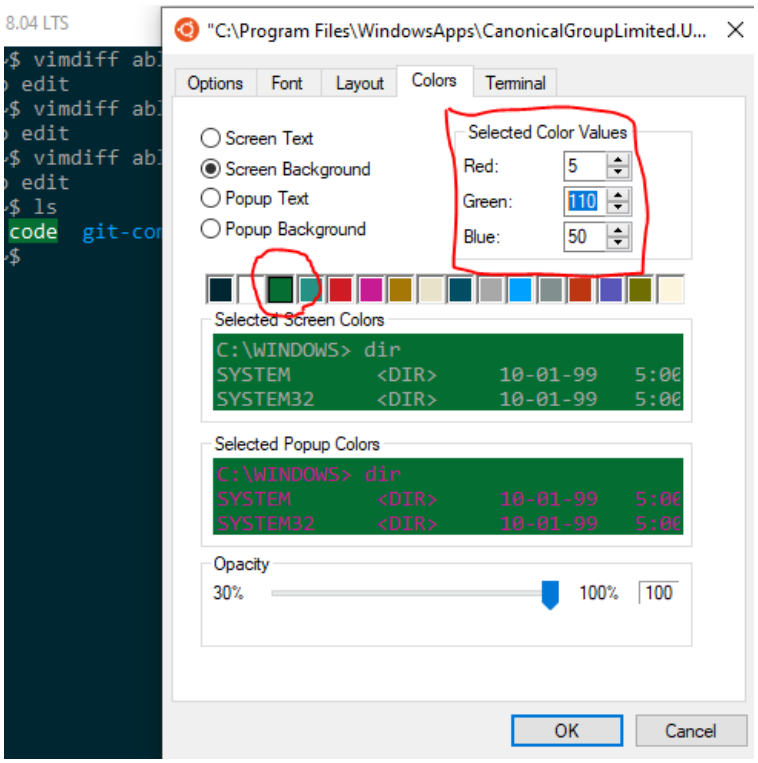
1.3.3 Fix Ubuntu terminal colors - Manually through properties

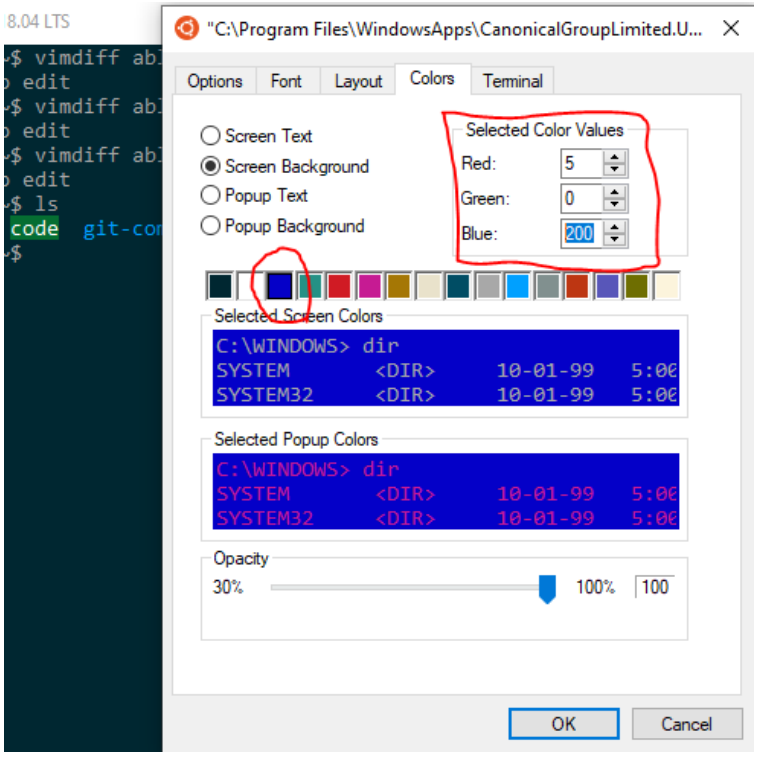
1. Open up the Ubuntu app.
2. Right-click on the top bar which says **Ubuntu 18.04 LTS** besides the icon logo. Press **Properties**. Go to the **Color** tab.
3. It is a weird color tool. When pressing **Screen Text** and **Screen Background**, observe which colors that are highlighted and note it down. These must be selected just before pressing **OK** later on.
   - These two selections are the main colors. The main background and main text. The selected ones, when pressing **OK** becomes the main colors.
   - One have to be careful of changing the colors as it is hard to reset them later on.
4. Lets say tha you want to change this green highlight (Screen Background). Press **Screen Background** and observe which color that is highlighted and thereby is the main color. Press the same green color as the one you want to change.



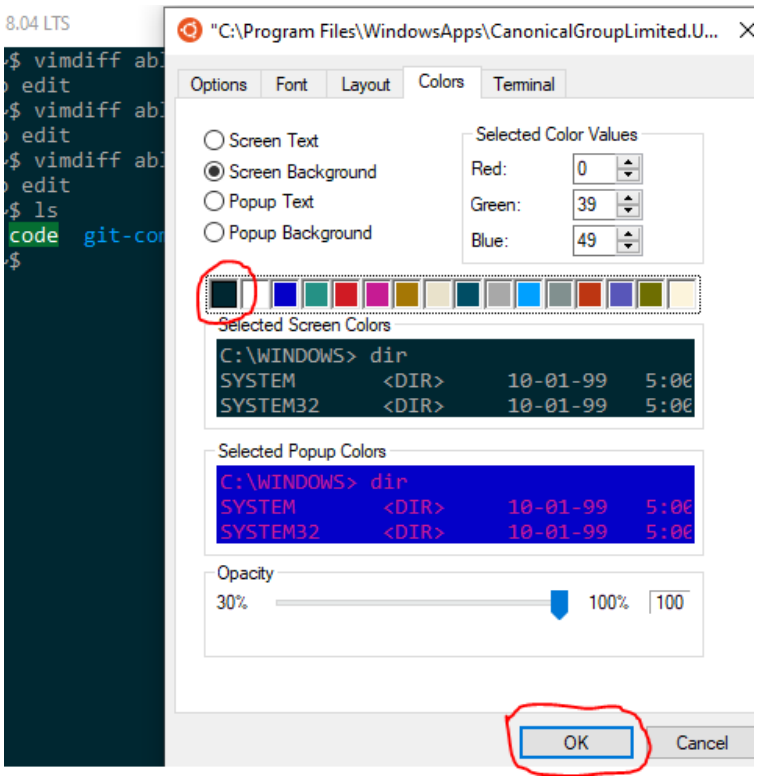


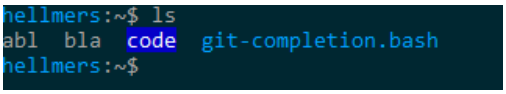
5. As the color you want to change is highlighted, change the RGB values to what you want to have instead.



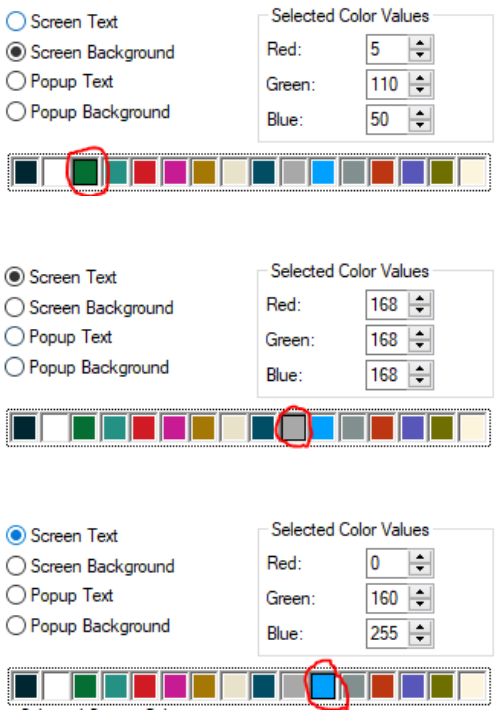


6. Press the previously highlighted (main color) and then press **OK**. Then the color have been changed.


8.04 LTS
$ vimdiff abl
edit
$ vimdiff abl
edit
$ vimdiff abl
edit
$ ls
code git-com
$
"C:\Program Files\WindowsApps\CanonicalGroupLimited.U...
Options
Font
Layout
Colors
Terminal
Screen Text
Screen Background
Popup Text
Popup Background
Selected Color Values
Red: 0
Green: 39
Blue: 49
Selected Screen Colors
C:\WINDOWS> dir
SYSTEM <DIR> 10-01-99 5:00
SYSTEM32 <DIR> 10-01-99 5:00
Selected Popup Colors
C:\WINDOWS> dir
SYSTEM <DIR> 10-01-99 5:00
SYSTEM32 <DIR> 10-01-99 5:00
Opacity
30%
100%
100
OK
Cancel


hellmers:~$ ls
abl bla code git-completion.bash
hellmers:~$

Here are some of my colors:



1.3.4 Fix vimdiff colors

The default colors of `vimdiff` can be really bad because of the translation from 16-bit colors to 256-bit colors.

Code can be found here:

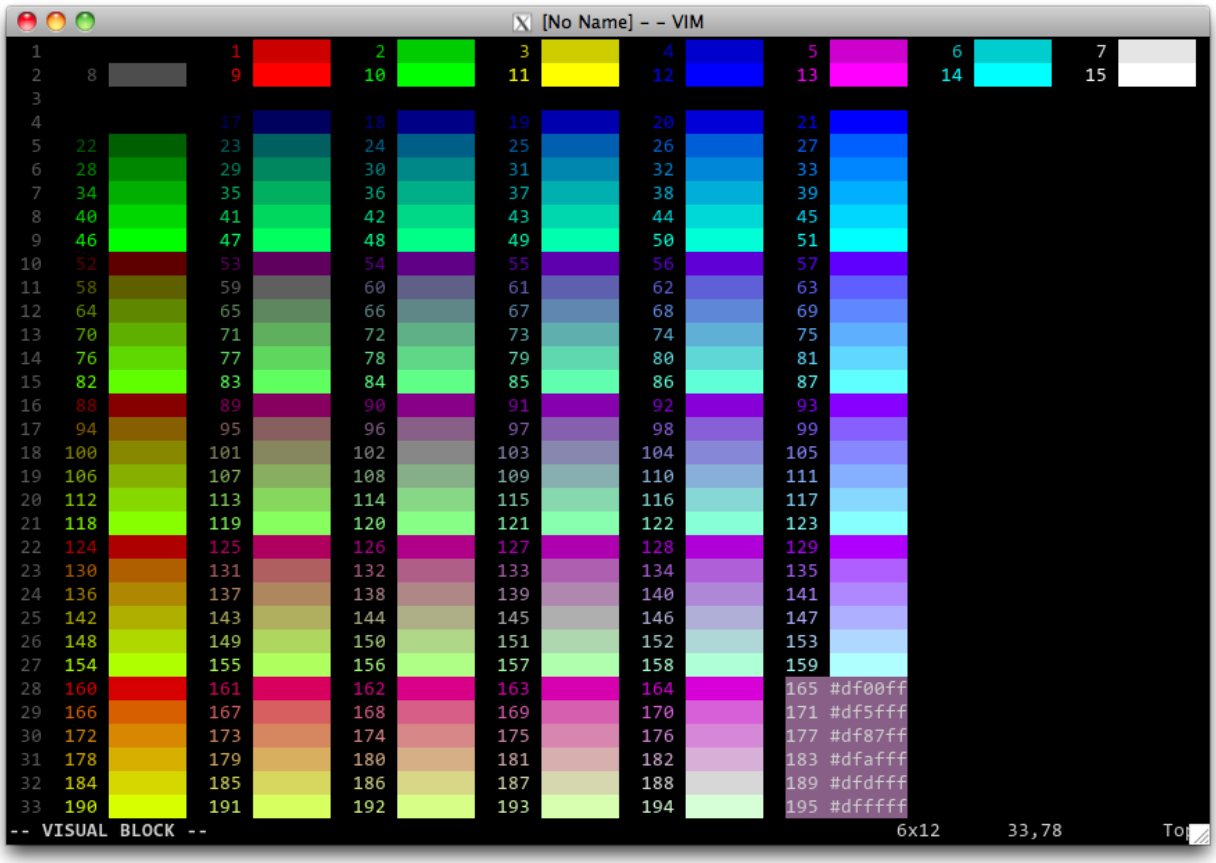
`https://github.com/robinhellmers/computer_setup/`

This can be fixed to something like this instead:

1. Create the `~/.vim/color/` directory.
2. Create a file `mycolorscheme.vim` in `~/.vim/color/`.
3. Paste this into the file (see Github):

```
highlight DiffAdd    cterm=bold ctermfg=15 ctermbg=22 gui=none guifg=bg guibg=Red
highlight DiffDelete cterm=bold ctermfg=15 ctermbg=88 gui=none guifg=bg guibg=Red
highlight DiffChange cterm=bold ctermfg=15 ctermbg=17 gui=none guifg=bg guibg=Red
highlight DiffText   cterm=bold ctermfg=15 ctermbg=130 gui=none guifg=bg guibg=Red
```

- `ctermfg` = foreground/text color
- `ctermbg` = background/highlight color
- Values given by xterm256 color table. This table might not correspond exactly to what you see on screen. Thereby it is better to print them out manually.



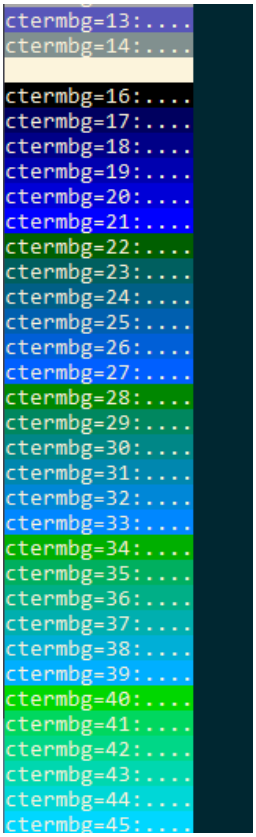
(a) Create a file `color_demo.vim` anywhere.

(b) Paste this into the file (see Github):

```
let num = 255
while num >= 0
    exec 'hi col_'.num.' ctermbg='.num.' ctermfg=white'
    exec 'syn match col_'.num.' "ctermbg='.num.':...." containedIn=ALL'
    call append(0, 'ctermbg='.num.':....')
    let num = num - 1
endwhile
```

(c) Open it up with `vim color_demo.vim` and then use the command `:so color_demo.vim`.

(d) This shows the background colors with corresponding values. Use **Page Up** and **Page down** to go through it.



4. You can continuously edit `~/.vim/colors/mycolorscheme.vim` and see the updates of the colors while still in `vimdiff` by using the command `:colo mycolorscheme`.

5. Now we are going to set this scheme permanently for `vimdiff`. Create a file in your home directory `~/.vimrc`.

6. Paste this into the file (see Github):

```
if &diff
    colorscheme mycolorscheme
endif
```

7. Now the custom color scheme should be applied every time you open `vimdiff`.

2 Virtual Machine Setup

2.1 Installation VirtualBox & Ubuntu 18.04

1. Download .iso file of Ubuntu 18.04.
2. Download and install VirtualBox.
3. Create new virtual machine.
   (a) Version: Ubuntu (64-bit); If not showing 64-bit, enable Virtual Machine in BIOS of host machine.
   (b) Next. Memory 4-5 GB if total 8 GB.
   (c) Next. Select `Create a virtual hard disk now`.
   (d) Create. Select `VDI`.
   (e) Next. Select `Dynamical`.
   (f) Next. 20-40 GB size. More towards 40 GB.
   (g) Create. Wait on creating storage. Done.
4. Settings of virtual machine.
   - → System → Motherboard; Memory still 4-5 GB.
   - → System → Motherboard; Enable I/O APIC
   - → System → Processor; 2 CPUs if total of 4 CPUs.
   - → Display → Screen; Max graphics memory.
5. Start virtual machine. Should ask for `start-up disk` where you `add` the .iso file in `Optical Disk Selector`. If not showing up follow following:
   (a) Go to settings → Storage.
   (b) Mark sub-group to `Controller: IDE`.
   (c) Under Attributes → Optical Drive; Press the circular button to the right.
   (d) Select `Choose/Create a Virtual Optical Disk...`
   (e) Add the .iso file.
   (f) Start virtual machine.
6. Choose to install Ubuntu.
7. Follow the steps and in one of the steps choose `Erase disk and install Ubuntu`. As this is a virtual machine, nothing will be erased on the host computer.

2.2 VirtualBox Extra Setup

2.2.1 Full-screen

1. Start virtual machine.
2. Press `Devices` drop down list in the virtual box window. That is, not inside the virtual machine itself.
3. Press `Insert Guest Additions CD image...`
4. A popup in the virtual machine should show: `... contains software intended to be automatically started. Would you like to run it?` and choose `Run`.
5. Resize the window a little and it will be full-screen.

2.2.2 Shared clipboard

1. In virtual box settings go to `General->Advanced` and select `Bidirectional` for `Shared Clipboard:`
2. Start virtual machine and see if it is working. If not, continue
3. Press `Devices` drop down list in the virtual box window. That is, not inside the virtual machine itself. Press `Insert Guest Additions CD image...`
4. If an error occurs do the following and then redo it
   - Unmount VBoxGuestAdditons by `Devices->Optical Drives->Remove disk from virtual drive`.
5. Reboot the virtual machine.
6. If it is not working, continue
7. Download and install *Extension pack* from virtual box.
8. Reboot the virtual machine.
9. If it is not working, try unmount and mount guest additions again.

2.2.3 Network setup for server and client IPv4 addresses

When starting virtual machine: `Ctrl + Alt + T` for terminal.
Write: `ip addr show`, check wether ip-address is something like `192.11.1.24` and not `10.0.1.1`.
If something with `10.(...)`, then it is a local IPv4 address and not one from the DHCP of the router.

Solution: Turn off virtual machine. Go to → Settings → Network and in `Attached to:` choose `Bridged Adapter` instead of probably NAT.

Start virtual machine and check if IPv4 address have changed to something like `192.(...)`.
If you open up a web-browser and don't get a connection, more settings have to be changed.
This probably depends on the virtual machine giving the router one MAC address and the host computer giving another.

Solution: Turn off virtual machine. Go to → Settings → Network and expand `Advanced`. Remove the MAC address. Then go to the host computer in Windows 10 and open `CMD`. Write: `ipconfig /all` and look for the MAC address of the host machine, probably named something like

```
Physical Address ..........................................  2C-F0-AF-73-2A-6C
```

Input this instead of the old removed MAC address and save. This probably makes you unable to use internet on the host machine instead which one will have to sacrifice.

2.3 VM Ubuntu installations

Everything in section 1.2 should be done.

2.3.1 Visual Studio Code

Install VS Code:

```
sudo snap install --classic code
```

3 Python

3.1 Python3 in Visual Studio Code

`https://stackoverflow.com/questions/50993566/vscode-there-is-no-pip-installer-available-in-the-selected`

4 Extra

4.1 Vim Settings

Create `/.vimrc`.

In order to permanently have numbering in vim/vi/vimdiff, add `:set number` into `.vimrc`.