Computer Setup Programming

Robin Hellmers

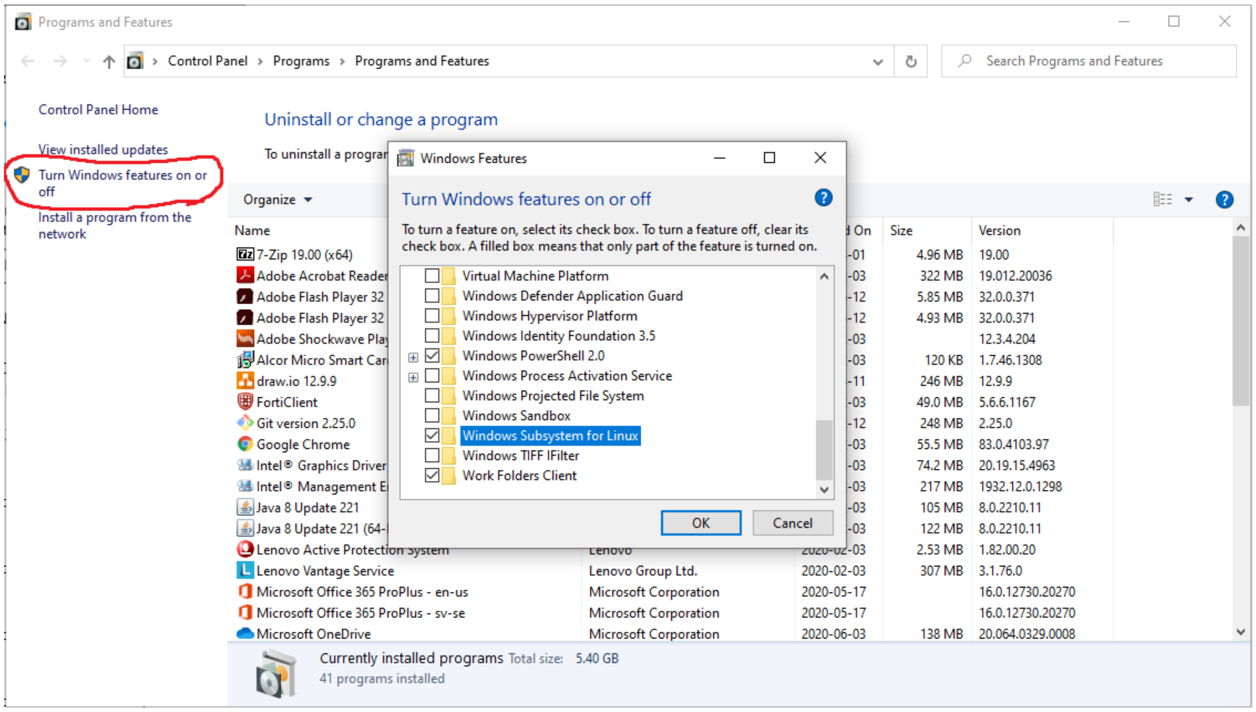
July 14, 2020

Contents

1 Ubuntu App - Windows Subsystem for Linux

1.1 Installation Ubuntu WSL with Visual Studio Code

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



Programs and Features

Control Panel › Programs › Programs and Features

Search Programs and Features

Control Panel Home

View installed updates

Turn Windows features on or off

Install a program from the network

Uninstall or change a program

To uninstall a program

Organize ▾

Name		d On	Size	Version
7-Zip 19.00 (x64)		-01	4.96 MB	19.00
Adobe Acrobat Reader		-03	322 MB	19.012.20036
Adobe Flash Player 32		-12	5.85 MB	32.0.0.371
Adobe Flash Player 32		-12	4.93 MB	32.0.0.371
Adobe Shockwave Play		-03		12.3.4.204
Alcor Micro Smart Car		-03	120 KB	1.7.46.1308
draw.io 12.9.9		-11	246 MB	12.9.9
FortiClient		-03	49.0 MB	5.6.6.1167
Git version 2.25.0		-12	248 MB	2.25.0
Google Chrome		-03	55.5 MB	83.0.4103.97
Intel® Graphics Driver		-03	74.2 MB	20.19.15.4963
Intel® Management E		-03	217 MB	1932.12.0.1298
Java 8 Update 221		-03	105 MB	8.0.2210.11
Java 8 Update 221 (64-		-03	122 MB	8.0.2210.11
Lenovo Active Protection System	Lenovo	2020-02-03	2.53 MB	1.82.00.20
Lenovo Vantage Service	Lenovo Group Ltd.	2020-02-03	307 MB	3.1.76.0
Microsoft Office 365 ProPlus - en-us	Microsoft Corporation	2020-05-17		16.0.12730.20270
Microsoft Office 365 ProPlus - sv-se	Microsoft Corporation	2020-05-17		16.0.12730.20270
Microsoft OneDrive	Microsoft Corporation	2020-06-03	138 MB	20.064.0329.0008

Currently installed programs Total size: 5.40 GB
41 programs installed

Windows Features

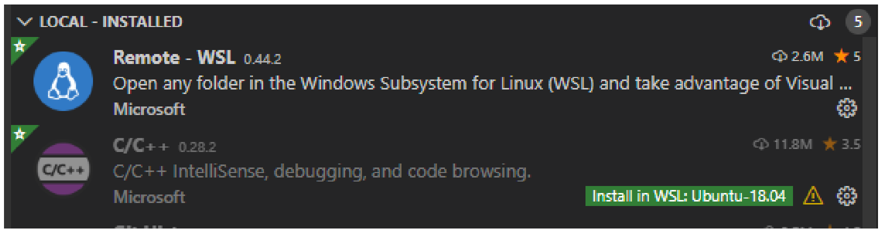
Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ☐ Virtual Machine Platform
- ☐ Windows Defender Application Guard
- ☐ Windows Hypervisor Platform
- ☐ Windows Identity Foundation 3.5
- ☑ Windows PowerShell 2.0
- ☐ Windows Process Activation Service
- ☐ Windows Projected File System
- ☐ Windows Sandbox
- ☑ Windows Subsystem for Linux
- ☐ Windows TIFF IFilter
- ☑ Work Folders Client

OK Cancel

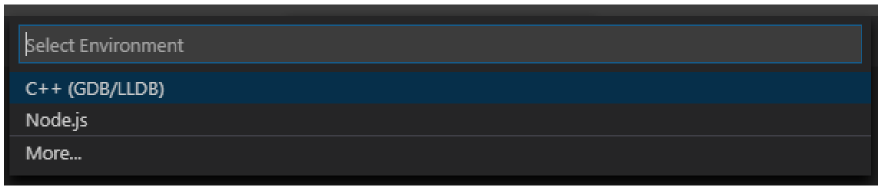
3. Restart computer.
4. Run `Ubuntu 18.04 LTS` and let it install. Might have to press `enter` after a while.
5. Create user with password.
6. Install `gcc` and `gdb` on the Windows Subsystem for Linux (WSL).
7. Create a new folder in the WSL where you create a standard main file for a C program. New folder is necessary for Visual Studio Code to realise that there is a C compiler to setup later on.
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`.
11. Right-click the `Ubuntu 18.04` and press `Connect to WSL`. A new window will appear with some connection to the WSL.
12. Press on the extension icon the left in the new window. Press `Install in WSL: Ubuntu-18.04` button on the `C/C++` extension.


LOCAL - INSTALLED

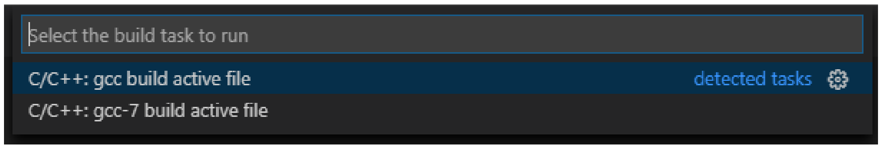
Remote - WSL 0.44.2
Open any folder in the Windows Subsystem for Linux (WSL) and take advantage of Visual ...
Microsoft

C/C++ 0.28.2
C/C++ IntelliSense, debugging, and code browsing.
Microsoft

13. By now it might prompt that you have to reload the window. Press that button.
14. Open up the folder you created the main C file in. `File->Open Folder...`
15. Open up the main C file in the file explorer.
16. Press `F5` for it to start the first compile and debug.
17. You will be prompted to choose what to use. First click `C++` and then `gcc` (not `gcc-7`).



18. Down at the `Output` and `Terminal`, press the three dots `...` and choose `Debug Console` in which one can run the standard `gdb` commands.
19. To just compile or build and not debug, press `Ctrl+Shift+B`.
20. You will be prompted to choose compiler, choose `gcc` (not `gcc-7`).


Select the build task to run
C/C++: gcc build active file
detected tasks
C/C++: gcc-7 build active file

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`

1.2.4 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

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:

```bash
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:

```bash
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:

```bash
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)
```

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-selecte