Setting up Ubuntu

This is an addendum to PS0 for getting Ubuntu Linux setup. You must do this to be able to do the projects. If you already have Ubuntu or one of its derivatives (like Linux Mint), you can skip to Section 2. If you are using a different distribution, proceed with caution; there may be some differences.

1 Installing Ubuntu

You will need to install Ubuntu. There are several options available to you.

1.1 VirtualBox

Ubuntu has directions for getting setup with VirtualBox at `https://ubuntu.com/tutorials/how-to-run-ubuntu-desktop-on-a-virtual-machine-using-virtualbox`. First, you will need to download an Ubuntu image from `https://ubuntu.com/download/desktop` and VirtualBox from `https://www.virtualbox.org/wiki/Downloads`. Then install VirtualBox.

Now create a new virtual machine. Make sure that you set the type to Linux and the version to Ubuntu (64-bit). When prompted to select a start-up disk, select the `.iso` file you downloaded.

1.2 WSL (Windows)

You can use Windows Subsystem for Linux (WSL) First, you will need to install WSL. Open a terminal and type

```
wsl --install -d Ubuntu
```

to install WSL with an Ubuntu shell. Alternately, you can download WSL manually from `https://docs.microsoft.com/en-us/windows/wsl/install`. Finally, you can alternately download the Ubuntu app from the Microsoft Store which sets up WSL with Ubuntu for you.

Unfortunately, WSL does not include the systems for displaying graphics or playing audio. We will need to setup an X server to display graphics and PulseAudio for playing audio.

You can install vcxsrv from `https://sourceforge.net/projects/vcxsrv/`. After it is installed, run XLaunch to start the X server. This will allow you to run GUI apps on Linux with WSL. **If the X server is closed (such as because you restarted your computer) you will need to restart the X server to run your GUI apps**.

You can acquire compiled PulseAudio binaries for Windows at `https://www.freedesktop.org/wiki/Software/PulseAudio/Ports/Windows/Support/`. The zip archive has compiled binaries, so you may need to override your browser's safety manager and give permission to download the archive. Once you have downloaded the zip archive, unzip it into the folder of your choice.

You will need to edit two of the PulseAudio config files. In the `etc/pulse/default.pa` file, add `record=0` to the end of line 42 so it reads

```
load-module module-waveout sink_name=output source_name=input record=0
```

This will disable recording; you don't need it and the Windows security manager doesn't like it. Afterwards, uncomment line 61 and add `auth-ip-acl=127.0.0.1;172.16.0.0/12` to the end.

```
load-module module-native-protocol-tcp auth-ip-acl=127.0.0.1;172.16.0.0/12
```

`127.0.0.1` is the IP address for your machine and 172.16.0.0/12 is the default IP address of the WSL virtual machine. This will give permission to both machines to connect to PulseAudio.

In the `etc/pulse/daemon.conf` file, uncomment line 39 and change the idle time to -1.

```
exit-idle-time = -1
```

A non-negative number is the timeout duration (in seconds). If no audio applications are connected for this long, the PulseAudio server will automatically shutdown. By setting it to a negative number, it will stay open until you explicitly close it.

At this point, you can now run the `bin/pulseaudio.exe` program to start the audio server. You will need to start this before running any audio applications on Ubuntu.

We also need to set a couple of things in our `.bashrc` file so that Ubuntu knows where to find our X-server and PulseAudio. Start WSL (which you can do by typing `wsl` at the terminal) and make sure you are in your home directory (`cd ~`). Edit your `.bashrc` file so that it contains

```
export HOST_IP="$(ip route|awk '/^default/{print $3}')"
export DISPLAY="$HOST_IP:0.0"
export PULSE_SERVER="tcp:$HOST_IP"
```

Then run `source ~/.bashrc` to reload the environment variables.

1.3 SSH

You can use an `ssh` client such as PuTTY to connect to a Linux server (such as `cs.uml.edu`). Note that you must connect to the VPN using the directions at `https://www.uml.edu/IT/Services/Get-connected/Remote-Access/` to ssh into the server from off campus. Be sure to use the -X option to enable X11 tunneling. If you are on Windows, you will also need to install an X-server and PulseAudio as in Section 1.2

2 Validating Setup

Once you have Ubuntu installed, we want to make sure that everything is setup correctly. First, make sure everything is up-to-date by running

```
sudo apt-get update
```

Afterwards, run

```
sudo apt-get install x11-apps
xclock
```

This will install some simple graphical programs and then run one of them. If everything is setup correctly, you should see a simple analog clock. If the clock fails to display, go back to Section 1 and make sure that you did everything correctly.

3 Configuring Ubuntu

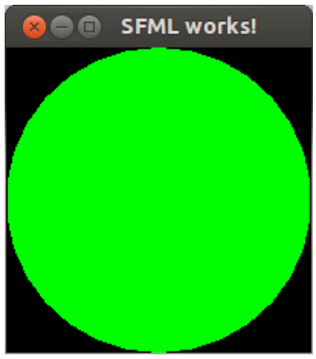
Now, we will setup some other tools. We'll be using `cpplint.py` later this semester to validate the formatting of our programs. This will require Python to be installed. We'll also use the `doxygen` tool to generate documentation for our projects.

```
sudo apt-get install doxygen
sudo apt-get install python
sudo apt-get install python-is-python3
sudo apt-get install python3-pip
pip install cpplint
```

We can install SFML by running

```
sudo apt-get install libsfml-dev
sudo apt-get install libpulse0
```

Which should setup the SFML and PulseAudio libraries for us. You can test it by doing the sample Hello World program as detailed in the PS0 assignment.



We will also be using several of the Boost libraries. Install them by using

```
sudo apt-get install libboost-test-dev
sudo apt-get install libboost-regex-dev
sudo apt-get install libboost-date-time-dev
```