Setting up a Development Environment on Linux

The aim of this guide is to show you how to install the GCC compiler onto a fresh version of Linux and how prepare it for using Orx.

There is a more concise version of the document here, for more experienced Linux users who just want the short overview.

We'll begin with a new Linux distribution. The one I've chosen is Lubuntu 18.04 Bionic. It comes in both 32-bit and 64-bit versions. The reason I chose this is because it is a reasonably small version of Ubuntu Linux, that ships with the bare essentials.

Note on choosing an architecture

If you have a 64-bit processor, you can install either the 32-bit or 64-bit version of Lubuntu. For 32-bit processors, you can only install the 32-bit version. Consider the audience for your game when choosing which architecture you will compile for. Do you intend for your game to only run on a 64-bit processor? Or will it be available as 32-bit to run on either 32 or 64-bit processors?

You can download the ISO here. There are many guides to installing Lubuntu onto a PC or as a Virtual Machine, so I'll leave that as an exercise for the reader.

Easier way to install gcc on Linux

A better way to install GCC on Ubuntu (at least) look to following this simpler guide: https://linuxize.com/post/how-to-install-gcc-on-ubuntu-20-04/

The compiler

Once you boot into the Linux desktop, open a Terminal window so that you can download the GCC compiler.

If you have a 64-bit processor and OS, do you want a system that can compile both 64-bit and 32-bit? If so, you'll need to get the multi-arch version of gcc and g++. If you only want the compiler for your current OS architecture, then you can install a regular gcc and g++.

Multi-arch GCC compiler

If you want the ability to compile both 32-bit and 64-bit binaries (and your OS is 64-bit), then you can choose to install the multi-arch version by doing this:

```
sudo apt install gcc-multilib g++-multilib
```

Regular GCC compiler

If you just want to compile for your current OS architecture (32-bit or 64-bit), then you can install with:

```
sudo apt install gcc g++
```

Installing Make

Regardless of what GCC you installed, you need to get `make` so that you can compile Orx:

```
sudo apt install make
```

Installing Orx dependent libraries

There are four development libraries that Orx uses that are not usually included in Linux by default.

Install these with:

```
sudo apt install libgl1-mesa-dev libxrandr-dev
```

Downloading and Compiling Orx

There is already a guide available to help you to clone Orx to your PC. Once you followed this, come back here after.

Once complete, you now have everything you need in order to compile Orx and your own Orx-based projects.

For the actual compiling of the Orx library, there is already a guide to help you using GCC/gmake (a mac version, but it is pretty much the same). You can follow that to help you compile the Orx library in either 32-bit or 64-bit (depending what CPU and OS you are running). But do return here for some extra notes on multi-arch compiling.

Extra notes on multi-arch compiling (32 and 64bit on the same machine)

Hopefully you have been successful compiling Orx. On a 64-bit system, you would have been able to compile 64-bit versions of the library and vice-versa for a 32-bit system. However, if you have installed the multi-arch GCC and G++, you can also compile for 32-bit on a 64-bit system.

In order to do that, you must first install the 32-bit versions of the development libraries that Orx depends on with:

sudo apt install libgl1-mesa-dev:i386 libxrandr-dev:i386

Now you will be able to compile with configs such as:

```
make config=debug32
```

Then you will be able compile 32 or 64-bit config at any time such as:

```
make config=release32
make config=release64
```

Read more about this at this post on the forum.