Chapter 1

Installation

1.1 Introduction

This is a guide for how to install Machine Programmer on Ubuntu Linux[**Ubuntu**]. Ubuntu Linux is perhaps the most popular Linux Distribution, also it is free to download and install (though donations are welcome).

If you have a different operating system such as those made by Microsoft or Apple, feel free to get it working in whatever method you use, and send corresponding documentation so it can be added as an option.

The version of Ubuntu Linux this tutorial will be refering to is 16.04, though at least for proprietary AMD drivers, you may have better luck on 14.04 or Fedora.

If you use a different version of Linux, I'm sure can adapt the package names to those which are suitable for your distribution. If you'd like to document how you did it then can forward it and will add your manual also.

1.2 Dependencies

As with any package the most difficult part is getting the dependencies installed.

needs autotools to compile

```
apt-get install autoconf automake libtool build-essential
```

1.2.1 OpenCL

The hardest part of compiling this is probably installing OpenCL. If you have that down, can simply:

```
./autogen.sh && ./configure && make && binary/programmer
```

Otherwise if that doesn't work, then you probably need OpenCL, so follow along below.

1.2.2 CPU OpenCL Installationdownload opencl header packages

If you'd like to take advantage of your CPU cores, or you don't have an OpenCL compatible GPU, then you'll have to install Portable Open Compute Library or POCL.

POCL has some dependencies on Ubuntu

```
apt-get install libhwloc-dev zlib1g-dev libclang-dev libx11-dev
   ocl-icd-dev
cmake opencl-headers
```

Then if you are sure you don't have GPU drivers, as installing this package may conflict with any GPU drivers you may install:

```
apt-get install ocl-icd-opencl-dev
```

then compile POCL based on it's instructions, if it's being difficult, can try the cmake part of the instructions, as cmake may give you additional dependencies.

Also note that you need to have likely the most recent OpenCL headers available, as of this writing, pocl-0.13 needs opencl 2.0 headers.

If you find that the dependencies I listed about are insufficient, then please email me with how you got it working. Otherwise once you have POCL can simply

```
./autogen.sh && ./configure && make && binary/programmer
```

1.2.3 GPU OpenCL Installation

For all GPU's the supported version of OpenCL may differ from the latest version. To avoid deprecation warnings and unexplained segmentation faults, can install the header files which are pertinent to your supported OpenCL version.

You can find out what version of OpenCL your hardware supports either from it's documentation, or by successfully installing OpenCL and then running the hello scripts, which will also give you information about your available OpenCL platforms.

For your convenience I've included zip files of the headers in library/. For example if your GPU supports OpenCL 1.1 then can install by doing:

```
cd library/
unzip OpenCL-Headers-opencl11.zip
mv OpenCL-Headers-opencl11 CL
sudo mv CL /usr/local/include/
```

(ARM) Mali GPU's (O-Droid XU3/4)

So far have tested it with the Mali OpenCL SDK, which works on the ODroid, an open-hardware heterogenous processing SoC board.

To run it need to do

```
apt-get install mali-x11 # and probably restart X server.
```

Because the standard opencl-headers provided by Ubuntu are 2.0+, and the Mali-T628 only supports up to 1.1 you may have to copy the CL folder from Mali SDK to /usr/local/include, to avoid a bunch of deprecation warnings and-or errors.

```
sudo cp -rv include/CL /usr/local/include
```

Strangely enough, in order to get it to compile, and to get it to run requires different versions of libOpenCL.so. However it seeems if POCL is installed, then can simply compile now with:

```
./autogen.sh && ./configure make && binary/programmer
```

Otherwise if POCL is Not installed the commands would be:

Mali's libOpenCL.so installation

In order to compile need the ones from Mali OpenCL SDK

After downloading that package, extract it and fix up platform.mk to reflect your current platform. For example:

```
CC:=arm-linux-gnueabihf-g++
AR=arm-linux-gnueabihf-ar
```

then in the Mali SDK folder compile the libOpenCL.so

```
cd lib/ && make
```

once you have a libOpenCL.so can put it into the machine-programmer's library/ folder.

```
cp lib/libOpenCL.so $MACHINE_PROGRAMMER_PATH/library/
```

```
./autogen.sh && ./configure LDFLAGS=-L./library && make &&
   binary/programmer
```

Intel GPUs

Intel includes onboard GPUs on a lot of motherboards, so chances are good that if you have an Intel CPU, that you may also have an intel GPU, which you can take advantage of using the well functioning open source beignet drivers.

```
apt-get install beignet beignet-dev beignet-opencl-icd
```

Note this does not work with the AMD hardware that I've tested it with.

Nvidia GPUs

Unfortunately there are no libre drivers for Nvidia that have OpenCL support, however there are proprietary drivers, which may work in certain cases. Note this means you can't use UEFI, common on modern laptops, you'll have to disable it in the bios. Also this could make breaking changes so I advise you backup your data before attempting to install proprietary drivers.

Before you begin, make sure your system is up to date;

```
sudo apt-get update && sudo apt-get upgrade && sudo apt-get
    dist-upgrade;
sudo apt-get autoremove #cleans up extra packages
```

There are a variety of versions of the nvidia drivers, this is because they are very finicky and they might not all work with your GPU card. For instance I had to try several, and spend several days testing, before I finally figured out which ones worked.

The best one for my GeForce GTX 960M, is unknown-361. I tried their recommended nvidia-367 and nvidia-370 those didn't work, and nvidia-340 didn't even show any picture on the screen (good thing I had ssh).

So I would advise you backup your system, and install ssh, so you could ssh tunnel into your computer and fix it in case the monitor stops working due to incompatible drivers.

Another good troubleshooting method when the screen is blank due to proprietary drivers, is to hold the shift-key during boot-up as this can let you enter the grub menu, where you can select "Advanced options" and use a recovery mode or different kernel.

Also may be a good idea to have a boot recovery disk or installation disk handy, because installing proprietary drivers may make the system unbootable.

At first reboot after installing the drivers I noticed that the computer turned itself off and on several times before loading properly, I'm guessing this is normal for prorpeitary drivers.

You may find it doesn't work well with the standard dev packs, something I was struggling with for days. But fortunately can install the open source ones from Intel.

```
apt-get install beignet-opencl-icd
```

You may also run into it not using the proprietary drivers, for instance if you do

```
lsmod |grep nvidia
```

it may give you nothing.

in this case may need to update-alternatives

```
sudo update-alternatives --config x86_64-linux-gnu_gl_conf
```

This however wont work if you don't have any working proprietary drivers.

I manged to get nvidia proprietary-drivers working for one or two boots, however I was not able to reliably reproduce it.

Other GPU's

You would have to see what is available for your platform, if you have success in getting OpenCL, then please write up a summary and email me.

1.3 Developing

If you would like to help with development, need some additional packages.

```
apt-get install clang-format git
```

1.4 Contact

Can email me at streondj@gmail.com for details.