Make and run UEFI applications on Ubuntu

Here you can setup and run first project named HelloWorld

Install Clion and enjoy autocomplete!

First of all install clion on ubuntu: To download CLion, visit the official website of JetBrains at https://www.jetbrains.com from your favorite web browser and go to Tools > CLion. Example downloaded file name is : CLion-2018.3.tar.gz, After downloading run :

```
$ sudo tar xzf CLion-2018.3.tar.gz -C /opt
    run clion by following command:
$ /opt/clion-2018.3/bin/clion.sh
```

After following all configurtions explained in https://linuxhint.com/install_jetbrains_clion_ubuntu/ open the vUDK2018 directory as a project in clion. and it would automatically create CmakeList.txt. if not download it from : https://github.com/Pouria-Empire/uefiCmake/blob/main/CMakeLists, and paste it in vUDK2018 directory. Now you can use autocomplete and fast coding by Ctrl+space and etc.

Install dependencies

In this article we use Clion and EDK2

Install dependencies

Based on Last document you can respectively run following commands:

```
$ sudo apt-get update
$ sudo apt-get install build-essential
$ sudo apt-get install git -y
$ sudo apt-get install autopoint -y
$ sudo apt-get install uuid-dev -y
$ sudo apt-get install cmake -y
$ sudo apt-get install libfreetype6-dev -y
$ sudo apt-get install libfontconfig1-dev -y
$ sudo apt-get install xclip -y
$ sudo apt-get install unifont -y
$ sudo apt-get install autotools-dev -y
$ sudo apt-get install autoconf -y
$ sudo apt-get install automake -y
$ sudo apt-get install iasl -y
$ sudo apt-get install qemu-system -y
$ sudo apt-get install nasm -y
$ nasm --version
```

```
    # NASM version 2.13.02
    // By default 'nasm' version 2.13 will be installed  in ubuntu 18.04 but
    // we need 'nasm' version 2.15. So, we need to install 'nasm' version 2.15 manually
    // [Ref] https://www.linuxfromscratch.org/blfs/view/cvs/general/nasm.html
$ wget https://www.nasm.us/pub/nasm/releasebuilds/2.15.05/nasm-2.15.05.tar.xz
$ tar -xf nasm-2.15.05.tar.xz --strip-components=1
$ ./configure --prefix=/usr && make
$ sudo make install
$ nasm --version
    # NASM version 2.15.05 compiled on DATE

$ gcc --version
    # gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
    // By default the GCC version is 7.5 in Ubuntu-18.04
$ sudo apt-get install g++-5 -y
$ sudo apt-get install gcc-5 -y
$ gcc --version
    # gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
    // The gcc is still in version 7.5, so we need to change the default gcc
    // [Ref] https://askubuntu.com/questions/26498/how-to-choose-the-default-gcc-and-g-v
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 10
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 20
$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-5 10
$ sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-5 20
$ sudo update-alternatives --install /usr/bin/cc cc /usr/bin/gcc 30
$ sudo update-alternatives --set cc /usr/bin/gcc
$ sudo update-alternatives --install /usr/bin/c++ c++ /usr/bin/g++ 30
$ sudo update-alternatives --set c++ /usr/bin/g++
$ gcc --version
    # gcc (Ubuntu 5.5.0-12ubuntu1) 5.5.0 20171010

$ sudo update-alternatives --install /usr/bin/python python /usr/bin/python3.6 10
$ python --version
    # Python 3.6.9
```

Now Start Coding

Git clone

Just clone the EDK2 git by following command:

```
`   $ cd $HOME      // Or any other directory you like to install EDK2
    $ git clone https://github.com/tianocore/edk2.git vUDK2018
    $ cd vUDK2018/
    $ git submodule update --init
    $ export EDK_TOOLS_PATH=$(pwd)/BaseTools
```

```
$ echo $EDK_TOOLS_PATH
    # $HOME/vUDK2018/BaseTools
$ make -C BaseTools/
    # ----------------------------------------------------------------------
    # Ran 301 tests in 0.756s
    # OK
    # make[1]: Leaving directory '/home/edk2/vUDK2018/BaseTools/Tests'
    # make: Leaving directory '/home/edk2/vUDK2018/BaseTools'
$ . edksetup.sh
```

build OVMF

For X64 processor architecture, go to vUDK2018 root directory and go to Conf then edit **target.txt** as follows:

```
$ gedit Conf/target.txt
    ACTIVE_PLATFORM       = OvmfPkg/OvmfPkgX64.dsc
    TARGET                = DEBUG
    TARGET_ARCH           = X64
    TOOL_CHAIN_TAG        = GCC5
```

Once you have modified Conf/target.txt, you can run the build command.

```
$ build
```

If successful, you should now have a OVMF.Fd file under the Build sub-directory.

Write first code

In vUDK2018 directory of cloned repository make a new directory called *HelloWorldPkg*. This is the main package we will working on that. In that directory create a directory called *HelloWorld* and that is our main module. In that module write this first C code: **HelloWorld.c**

```
#include <Uefi.h>
#include <Library/UefiLib.h>

EFI_STATUS
EFIAPI
UefiMain (
  IN EFI_HANDLE        ImageHandle,
  IN EFI_SYSTEM_TABLE  *SystemTable
  )
{
  Print(L"Hello, world...?\n");

  return EFI_SUCCESS;
}
```

And make **HelloWorld.inf**

```
[Defines]
  INF_VERSION                    = 0x00010005
  BASE_NAME                      = HelloWorld
  FILE_GUID                      = 869460ce-579c-4cb3-bbe4-613b576ebc22
  MODULE_TYPE                    = UEFI_APPLICATION
  VERSION_STRING                 = 1.0
  ENTRY_POINT                    = UefiMain

[Sources]
  HelloWorld.c

[Packages]
  MdePkg/MdePkg.dec
#  MdeModulePkg/MdeModulePkg.dec

[LibraryClasses]
  UefiApplicationEntryPoint
  UefiLib
#  PcdLib
```

Now change directory to ../HelloWorldPkg. In this directory you need to place two files called respectively **HelloWorldPkg.dec** and **HelloWorldPkg.dsc**. Here is **HelloWorldPkg.dsc**

```
[Defines]
  PLATFORM_NAME                  = HelloWorld
  PLATFORM_GUID                  = f67df959-c108-4755-a6df-6204a4719d22
  PLATFORM_VERSION               = 0.1
  DSC_SPECIFICATION              = 0x00010005
  OUTPUT_DIRECTORY               = Build/HelloWorld
  SUPPORTED_ARCHITECTURES        = IA32|X64|ARM|AARCH64|RISCV64
  BUILD_TARGETS                  = DEBUG|RELEASE|NOOPT
  SKUID_IDENTIFIER               = DEFAULT

[LibraryClasses]
  BaseLib|MdePkg/Library/BaseLib/BaseLib.inf
  BaseMemoryLib|MdePkg/Library/BaseMemoryLib/BaseMemoryLib.inf
  DebugLib|MdePkg/Library/BaseDebugLibNull/BaseDebugLibNull.inf
  DebugPrintErrorLevelLib|MdePkg/Library/BaseDebugPrintErrorLevelLib/BaseDebugPrintErrorLeve
  PcdLib|MdePkg/Library/BasePcdLibNull/BasePcdLibNull.inf
  PrintLib|MdePkg/Library/BasePrintLib/BasePrintLib.inf
  RegisterFilterLib|MdePkg/Library/RegisterFilterLibNull/RegisterFilterLibNull.inf
  UefiApplicationEntryPoint|MdePkg/Library/UefiApplicationEntryPoint/UefiApplicationEntryPoi
  UefiBootServicesTableLib|MdePkg/Library/UefiBootServicesTableLib/UefiBootServicesTableLib
```

```
  UefiLib|MdePkg/Library/UefiLib/UefiLib.inf
  UefiRuntimeServicesTableLib|MdePkg/Library/UefiRuntimeServicesTableLib/UefiRuntimeServices
  DevicePathLib|MdePkg/Library/UefiDevicePathLib/UefiDevicePathLib.inf

[LibraryClasses.common.UEFI_APPLICATION]
  MemoryAllocationLib|MdePkg/Library/UefiMemoryAllocationLib/UefiMemoryAllocationLib.inf
  DebugLib|MdePkg/Library/UefiDebugLibStdErr/UefiDebugLibStdErr.inf

[Components]
  HelloWorldPkg/HelloWorld/HelloWorld.inf
```

And here is **HelloWorldPkg.dec**

```
[Defines]
  DEC_SPECIFICATION              = 0x00010005
  PACKAGE_NAME                   = HelloWorldPkg
  PACKAGE_GUID                   = a1fc00e6-6e44-4d57-9f75-63048b2976c3
  PACKAGE_VERSION                = 0.0.1

[Includes]
  Include
```

Now just save them and create a directory called *Include*. Then go to vUDK2018 root directory and go to Conf then edit **target.txt** as follows:

```
    $ gedit Conf/target.txt
        ACTIVE_PLATFORM       = HelloWorldPkg/HelloWorldPkg.dsc
        TARGET                = RELEASE
        TARGET_ARCH           = X64
        TOOL_CHAIN_CONF       = Conf/tools_def.txt
        TOOL_CHAIN_TAG        = GCC5
```

Now go back to vUDK2018. and create an executable file named **start.sh**

```
#!/bin/sh
mkdir -p /tmp/qemu-hda/EFI/BOOT/
ln -sf $(pwd)/Build/"$1"/RELEASE_GCC5/X64/"$1".efi /tmp/qemu-hda/EFI/BOOT/BOOTX64.EFI
exec qemu-system-x86_64 \
    -net none\
    -bios ./Build/OvmfX64/DEBUG_GCC5/FV/OVMF.fd \
    -drive file=fat:rw:/tmp/qemu-hda
```

Now run the following commands:

```
$ build

$ ./start.sh HelloWorld
```

As you can run the BOOTX64.efi in EFI shell by type FS0: and go EFI then BOOT then run BOOTX64.efi

Well done!