


UEFI & EDK II Training

PLATFORM BUILD LAB - OVMF

tianocore.org

PLATFORM BUILD LABS

- ✱ Build a EDK II Platform using OVMF package 
- ✱ Run Ovmf using Qemu

BUILD OVMFPKG

Setup OvmfPkg to build and run w/ QEMU

Pre-requisites Ubuntu 20.04

Example Ubuntu 20.04

The following need to be accessible for building Edk2, From the terminal prompt (Cnt-Alt-T):

```
bash$ sudo apt install build-essential uuid-dev iasl git nasm python-is-python3
```

- build-essential - Informational list of build-essential packages
- uuid-dev - Universally Unique ID library (headers and static libraries)
- iasl - Intel ASL compiler/decompiler (also provided by acpica-tools)
- git - support for git revision control system
- nasm - General-purpose x86 assembler
- python-is-python3 - Ubuntu 20.04 python command is 'python3' but edk2 tools use 'python'

The following will install the QEMU for Intel X86 & 64 bit

```
bash$ sudo apt install qemu-system-x86-64
```

- Qemu – Emulation with Intel architecture with UEFI Shell

See Lab guide for Ubuntu 16.04 pre-requisites



ubuntu

Pre-requisites Clear Linux* Project

Example Using Clear Linux* Project

The following need to be accessible for building Edk2, From the terminal prompt (Cnt-Alt-T):

```
bash$ sudo swupd bundle-add devpkg-util-linux
```

Devpkg-util-linux – includes bundles for developer tools for writing “C” Applications included:
gcc, nasm, uuid, etc.

```
bash$ sudo swupd bundle-add kvm-host
```

Qemu – Emulation with Intel architecture with UEFI Shell



Create QEMU Run Script

1. Create a run-ovmf directory under the home directory

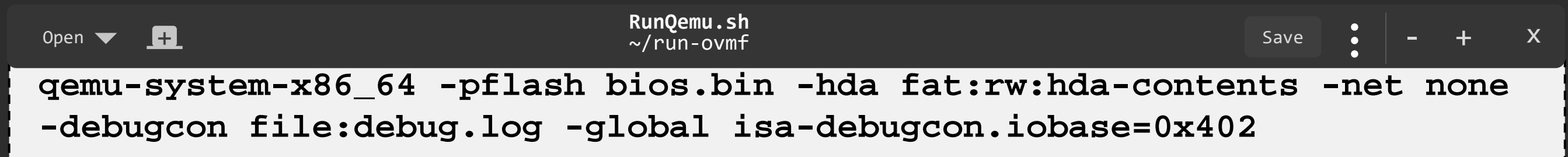
```
bash$ cd ~  
bash$ mkdir ~run-ovmf  
bash$ cd run-ovmf
```

2. Create a directory to use as a hard disk image

```
bash$ mkdir hda-contents
```

3. Create a Linux shell script to run the QEMU from the run-ovmf directory

```
bash$ gedit RunQemu.sh
```



```
RunQemu.sh  
~/run-ovmf  
qemu-system-x86_64 -pflash bios.bin -hda fat:rw:hda-contents -net none  
-debugcon file:debug.log -global isa-debugcon.iobase=0x402
```

4. Save and Exit

DOWNLOAD the EDK II Source

Open a terminal prompt and create a source working directory

```
bash$ mkdir ~/src  
bash$ cd ~/src  
bash$ mkdir edk2-ws
```

Internet Proxies – (company Firewall used for example)

```
bash$ export http_proxy=http://proxy-us.company.com:911  
bash$ export ftp_proxy=$http_proxy
```

Download edk2 source tree using Git

```
bash$ git clone -b Edk2Lab_22Q1 https://github.com/tianocore-training/edk2.git  
Bash$ git clone https://github.com/tianocore/edk2-libs.git
```


Download the Submodules and Checkout the Lab Branch

```
bash$ cd edk2  
bash$ submodule update -init  
bash$ cd ..
```

SETUP LAB MATERIAL

Lab_Material_FW.zip

DOWNLOAD LAB MATERIAL

Download the Lab_Material_FW.zip from :  [github.com](https://github.com/tianocore-training/Lab_Material_FW.git)
[Lab_Material_FW.zip](#)

OR

Use git clone to download the Lab_Material_FW

```
bash$ cd $HOME
bash$ git clone https://github.com/tianocore-training/Lab_Material_FW.git
```

Directory Lab_Material_FW will be created

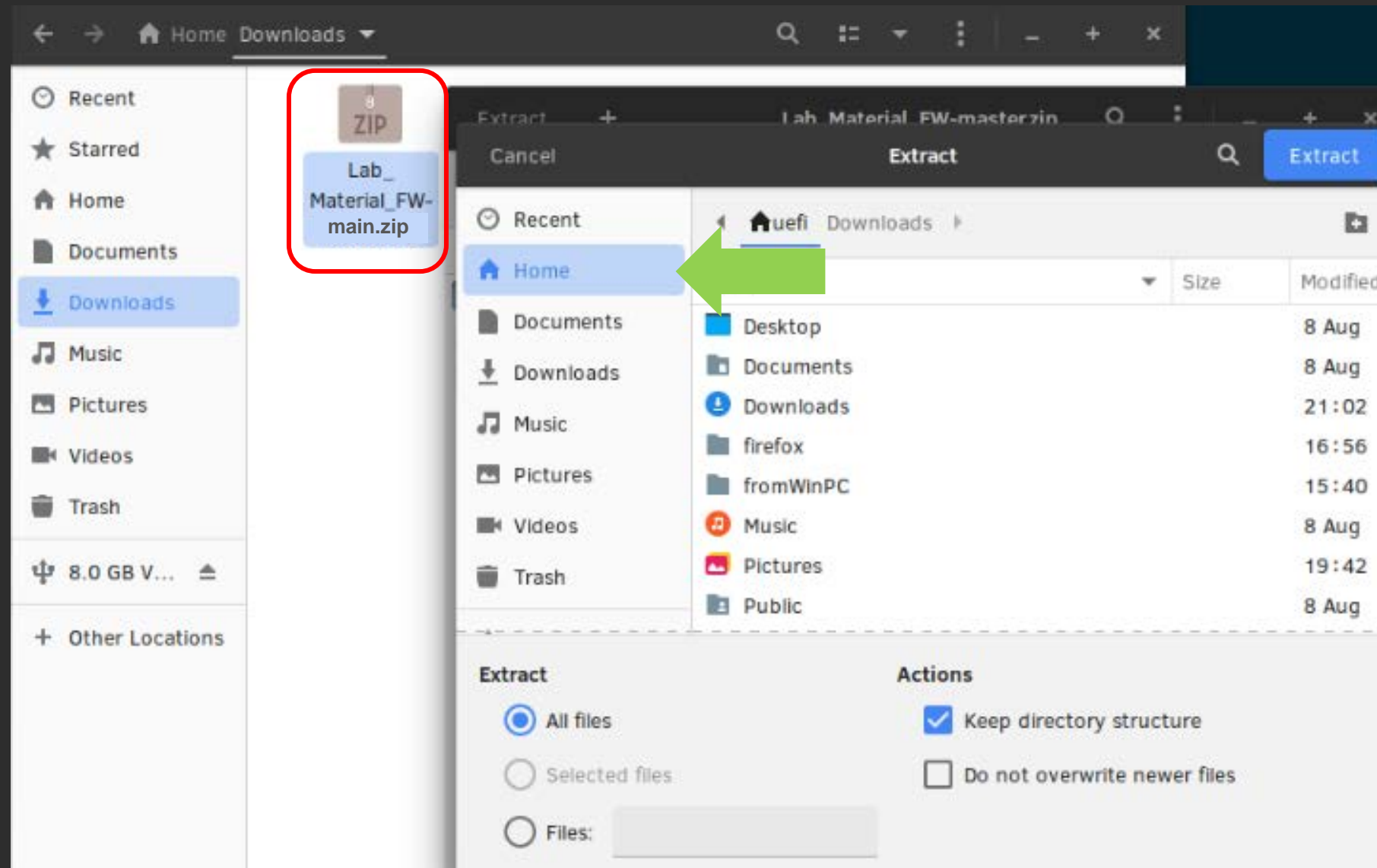
FW

- Documentation
- DriverWizard
- edk2-ws
- edk2Linux
- LabSampleCode

BUILD EDK II OVMF

-Extract the Source

1. Extract the Downloaded Lab_Material_FW-main.zip to Home (this will create a directory ~/FW)



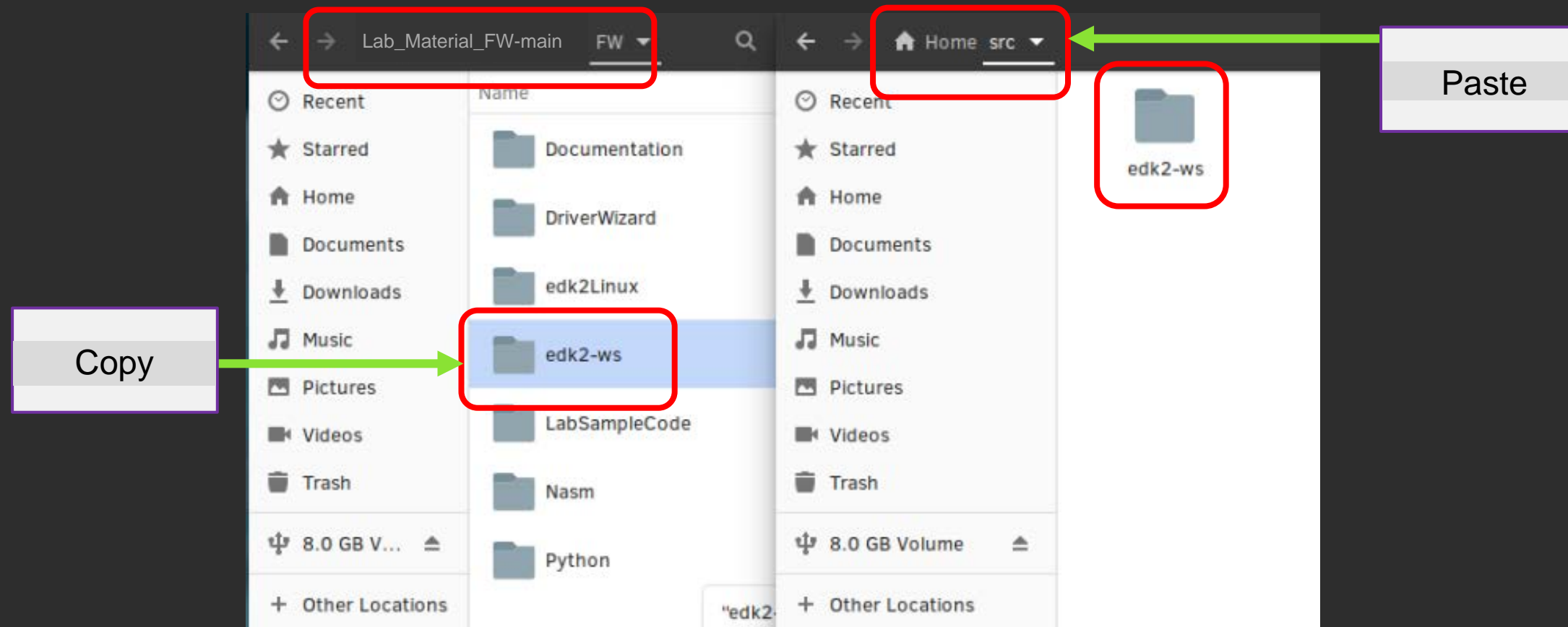
BUILD EDK II OVMF

- Copy the Source

2. Open a terminal prompt (Alt-Cnt-T)
3. Create a working space source directory under the home directory

```
bash$ cd ~src
```

4. From the FW folder, copy and paste folder “~.../FW/edk2-ws” to ~src



BUILD EDK II OVMF

- Building BaseTools

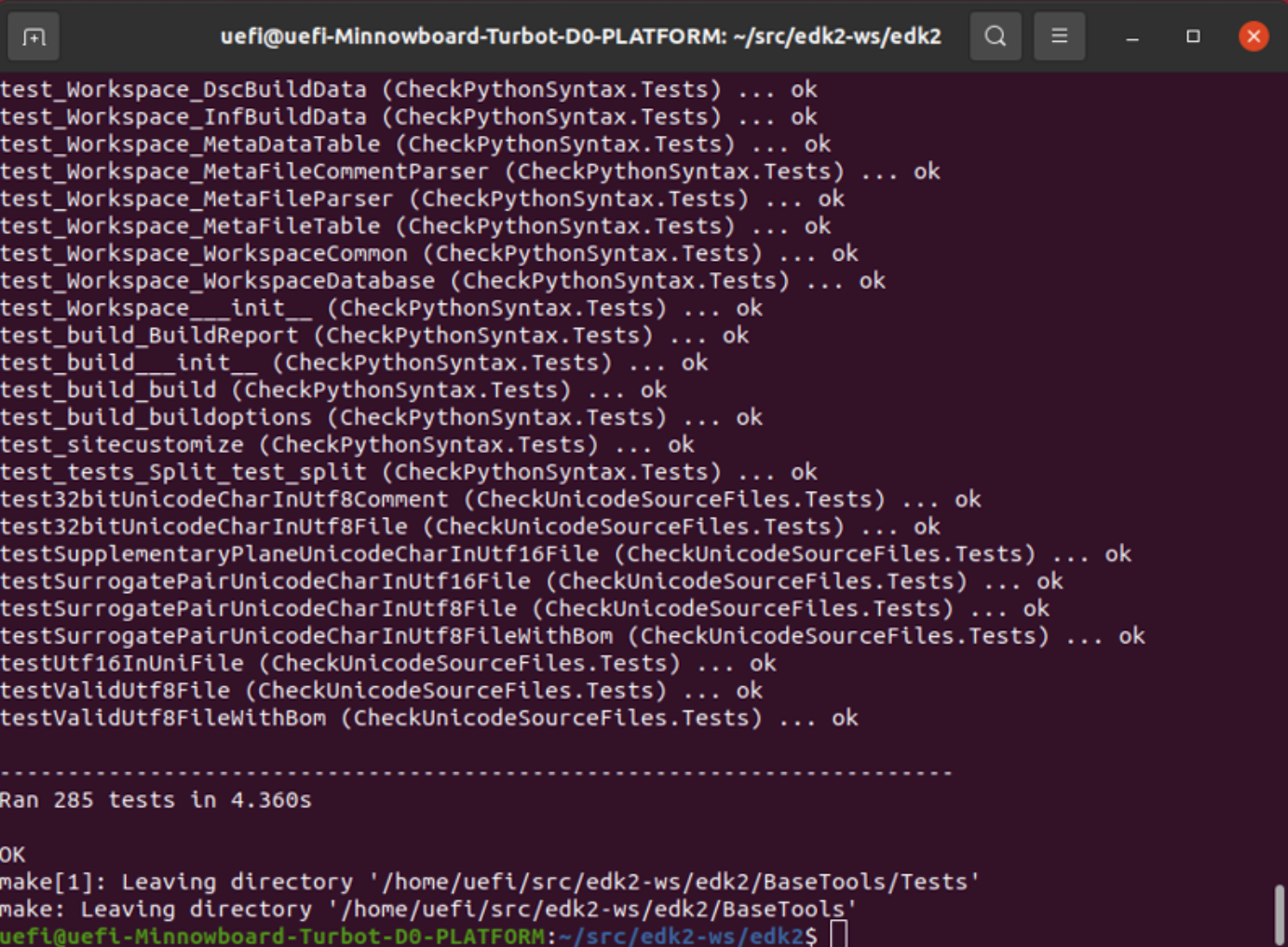
5. Export work space & platform path

```
bash$ cd ~src/edk2-ws
bash$ export WORKSPACE=$PWD
bash$ export PACKAGES_PATH=$WORKSPACE/edk2:$WORKSPACE/edk2-libc
```

6. Run Make

```
bash$ cd edk2
bash$ make -C BaseTools/
```

7. Make sure the tests pass OK



```
uefi@uefi-Minnowboard-Turbot-D0-PLATFORM: ~/src/edk2-ws/edk2
test_Workspace_DscBuildData (CheckPythonSyntax.Tests) ... ok
test_Workspace_InfBuildData (CheckPythonSyntax.Tests) ... ok
test_Workspace_MetaDataTable (CheckPythonSyntax.Tests) ... ok
test_Workspace_MetaFileCommentParser (CheckPythonSyntax.Tests) ... ok
test_Workspace_MetaFileParser (CheckPythonSyntax.Tests) ... ok
test_Workspace_MetaDataTable (CheckPythonSyntax.Tests) ... ok
test_Workspace_WorkspaceCommon (CheckPythonSyntax.Tests) ... ok
test_Workspace_WorkspaceDatabase (CheckPythonSyntax.Tests) ... ok
test_Workspace___init__ (CheckPythonSyntax.Tests) ... ok
test_build_BuildReport (CheckPythonSyntax.Tests) ... ok
test_build___init__ (CheckPythonSyntax.Tests) ... ok
test_build_build (CheckPythonSyntax.Tests) ... ok
test_build_buildoptions (CheckPythonSyntax.Tests) ... ok
test_sitecustomize (CheckPythonSyntax.Tests) ... ok
test_tests_Split_test_split (CheckPythonSyntax.Tests) ... ok
test32bitUnicodeCharInUtf8Comment (CheckUnicodeSourceFiles.Tests) ... ok
test32bitUnicodeCharInUtf8File (CheckUnicodeSourceFiles.Tests) ... ok
testSupplementaryPlaneUnicodeCharInUtf16File (CheckUnicodeSourceFiles.Tests) ... ok
testSurrogatePairUnicodeCharInUtf16File (CheckUnicodeSourceFiles.Tests) ... ok
testSurrogatePairUnicodeCharInUtf8File (CheckUnicodeSourceFiles.Tests) ... ok
testSurrogatePairUnicodeCharInUtf8FileWithBom (CheckUnicodeSourceFiles.Tests) ... ok
testUtf16InUniFile (CheckUnicodeSourceFiles.Tests) ... ok
testValidUtf8File (CheckUnicodeSourceFiles.Tests) ... ok
testValidUtf8FileWithBom (CheckUnicodeSourceFiles.Tests) ... ok

-----
Ran 285 tests in 4.360s

OK
make[1]: Leaving directory '/home/uefi/src/edk2-ws/edk2/BaseTools/Tests'
make: Leaving directory '/home/uefi/src/edk2-ws/edk2/BaseTools'
uefi@uefi-Minnowboard-Turbot-D0-PLATFORM:~/src/edk2-ws/edk2$
```



BUILD OVMF PLATFORM

BUILD EDK II OVMF

-Update Target.txt

What is OVMF?

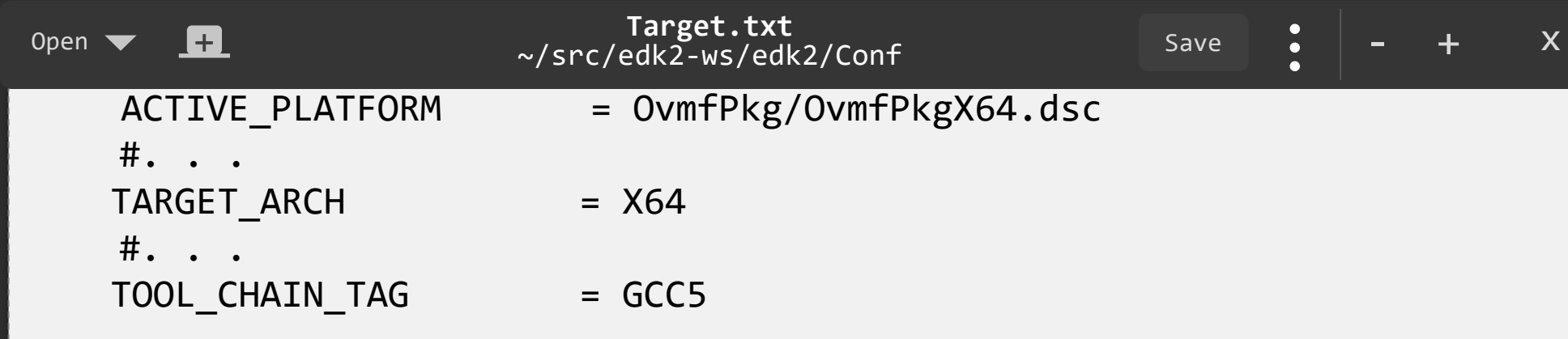
Open Virtual Machine Firmware - Build with edk2

```
bash$ cd ~/src/edk-ws/edk2
bash$ . edksetup.sh
```

```
uefi@clr-0~/src/edk2-ws/edk2 $ . edksetup.sh
Loading previous configuration from /home/uefi/src/edk2-ws/edk2/Conf/Build
WORKSPACE: /home/uefi/src/edk2-ws
EDK_TOOLS_PATH: /home/uefi/src/edk2-ws/edk2/BaseTools
CONF_PATH: /home/uefi/src/edk2-ws/edk2/Conf
uefi@clr-0~/src/edk2-ws/edk2 $
```

Edit the file Conf/target.txt

```
bash$ gedit Conf/target.txt
```



```
Open ▼ + Target.txt
~/src/edk2-ws/edk2/Conf Save ⋮ - + X
1. ACTIVE_PLATFORM = OvmfPkg/OvmfPkgX64.dsc
   #. . .
2. TARGET_ARCH = X64
   #. . .
3. TOOL_CHAIN_TAG = GCC5
```

Save and build

```
bash$ build -D ADD_SHELL_STRING
```

More info: [tianocore - wiki/OVMF](https://tianocore.org/wiki/OVMF)

BUILD EDK II OVMF

-Inside Terminal

```

uefi@uefi-Minnowboard-Turbot-D0-PLATFORM: ~/src/edk2-ws/edk2
Generate Region at Offset 0xF000
Region Size = 0x1000
Region Name = None
Generate Region at Offset 0x10000
Region Size = 0x10000
Region Name = None
Generate Region at Offset 0x20000
Region Size = 0xE0000
Region Name = FV
Generate Region at Offset 0x100000
Region Size = 0xC00000
Region Name = FV
GUID cross reference file can be found at /home/uefi/src/edk2-ws/Build/OvmfX64/DEBUG_GCC5/SECDEFV/SecDefV.xref
FV Space Information
SECFV [20%Full] 212992 (0x34000) total, 42688 (0xa6c0) used, 170304 (0x29940) free
PEIFV [23%Full] 917504 (0xe0000) total, 211112 (0x338a8) used, 706392 (0xac758) free
DXEFV [35%Full] 12582912 (0xc00000) total, 4419600 (0x437010) used, 8163312 (0x7c1ff0) free
FVMAIN_COMPACT [36%Full] 3440640 (0x348000) total, 1251760 (0x1319b0) used, 2188880 (0x216650) free
Done -
Build end time: 11:40:47, Apr.05 2022
Build total time: 00:06:02
uefi@uefi-Minnowboard-Turbot-D0-PLATFORM:~/src/edk2-ws/edk2$
uefi@uefi-Minnowboard-Turbot-D0-PLATFORM:~/src/edk2-ws/edk2$

Active Platform = /home/uefi/src/edk2-ws/edk2/OvmfPkg/OvmfPkgX64.dsc

```

Finished build

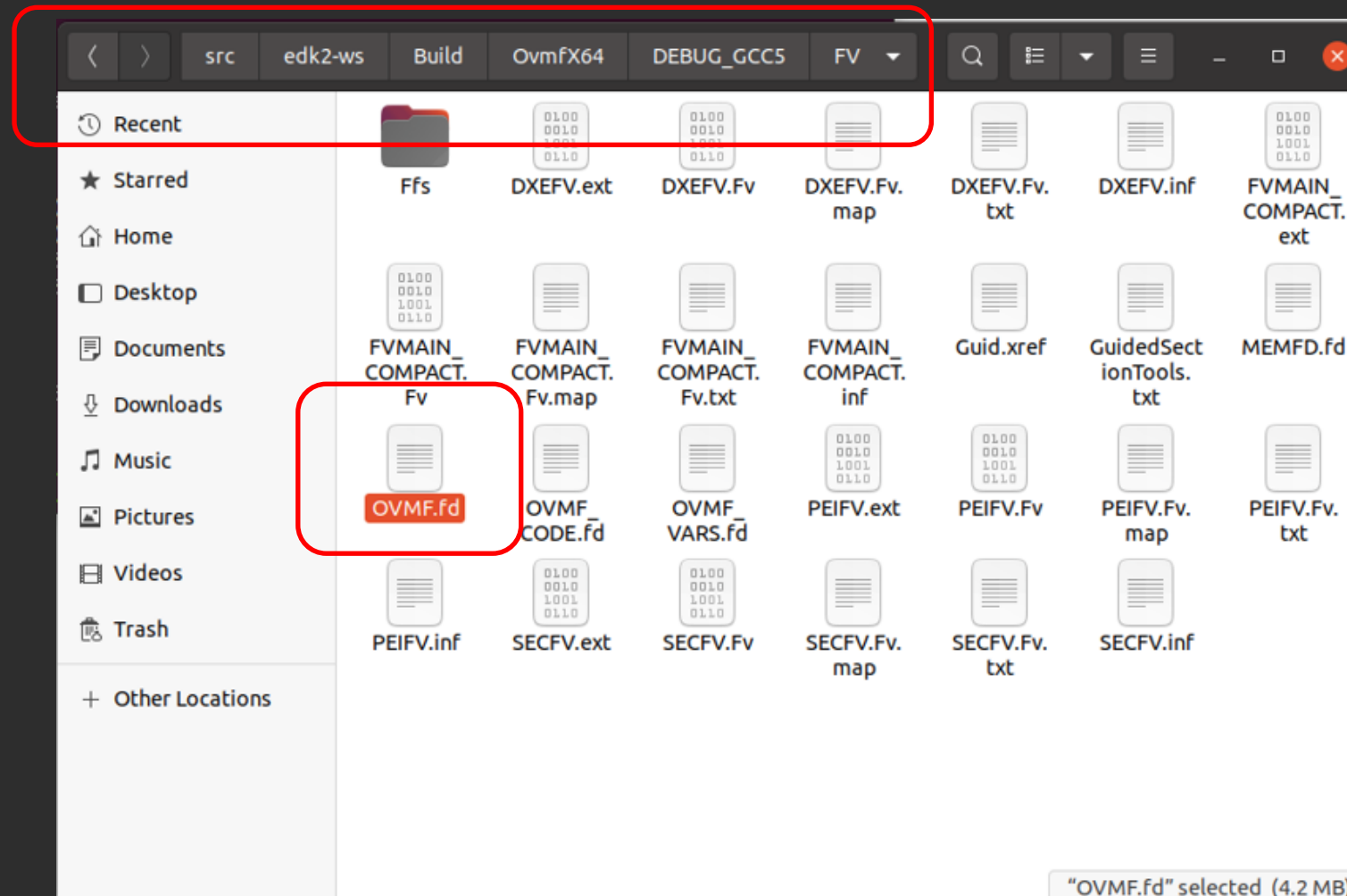
BUILD EDK II OVMF

-Verify Build Succeeded

OVMF.fd should be in the Build directory

- For GCC5 with X64, it should be located at

```
~/src/edk2-ws/Build/OvmfX64/DEBUG_GCC5/FV/OVMF.fd
```





Change to run-ovmf directory under the home directory

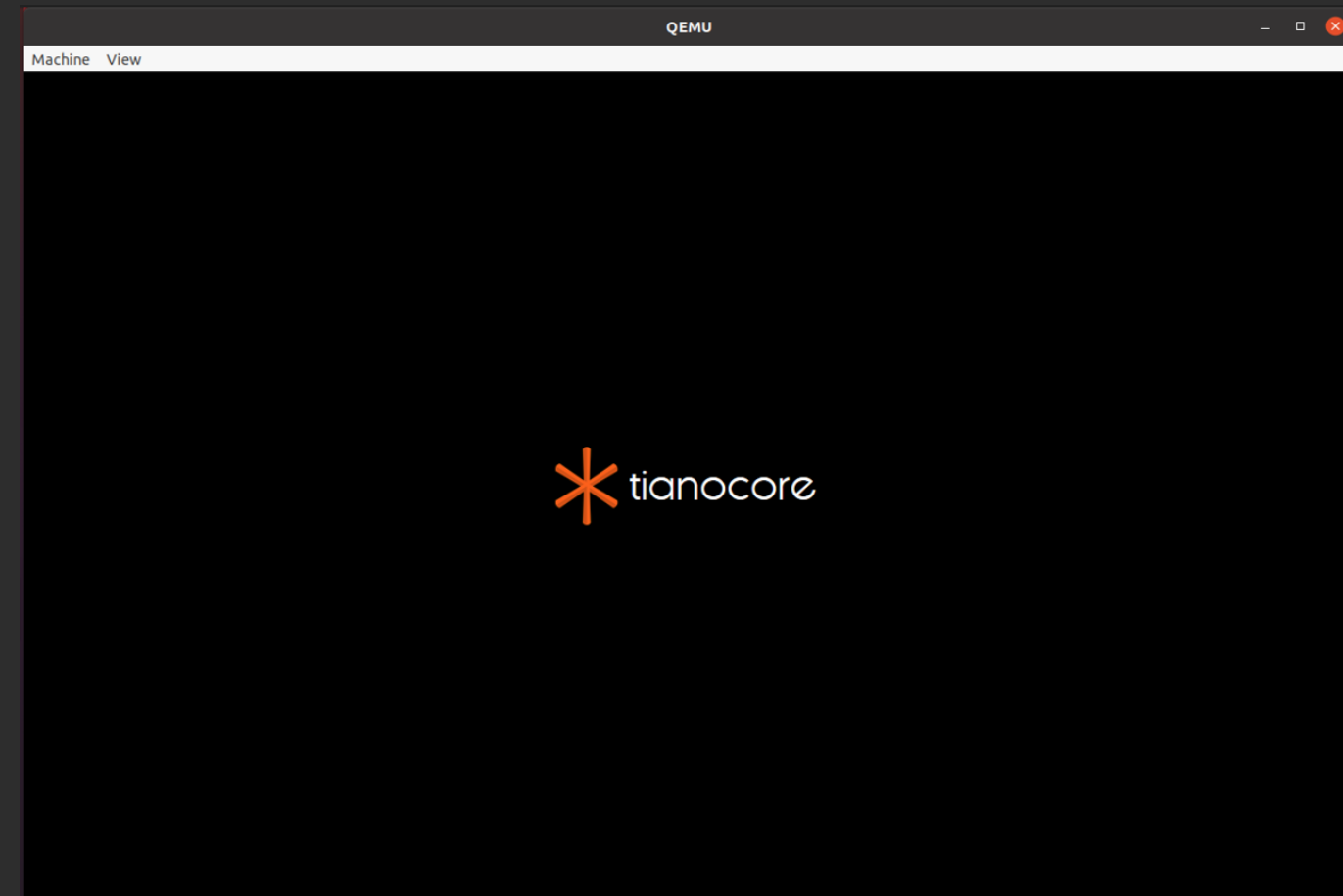
```
bash$ cd $HOME/run-ovmf
```

Copy the OVMF.fd BIOS image created from the build to the run-ovmf directory naming it bios.bin

```
bash$ cp ~/src/edk2-  
ws/Build/OvmfX64/DEBUG_GCC5/FV/OVMF.fd  
bios.bin
```

Run the RunQemu.sh Linux shell script

```
bash$ . RunQemu.sh
```



Show the UEFI Boot Variables

At the Shell Prompt:

Shell> FS0:

FS0:> BCFG Boot Dump

```
Desc      - UEFI BootManagerMenuApp
DevPath   - Fv (6D99E806-3D38-42C2-A095-5F4300BFD7DC) /FuFile (EEC25BDC-67F2-4D95-B
1D5-F81B2039D11D)
Optional- N
Option: 02. Variable: Boot0002
Desc      - UEFI Misc Device
DevPath   - VenHw (5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,000000000) /VenHw (6888A4AE-
AFCE-E84B-9102-F7B9DAE6A030,000000000)
Optional- Y
Option: 03. Variable: Boot0003
Desc      - UEFI Non-Block Boot Device
DevPath   - VenHw (5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,000000000) /VenHw (964E5B22-
6459-11D2-8E39-00A0C969723B,000000000)
Optional- Y
Option: 04. Variable: Boot0004
Desc      - UEFI BootManagerMenuApp
DevPath   - Fv (6D99E806-3D38-42C2-A095-5F4300BFD7DC) /FuFile (EEC25BDC-67F2-4D95-B
1D5-F81B2039D11D) /BootManagerMenuApp
Optional- Y
Option: 05. Variable: Boot0000
Desc      - UEFI Enter Setup
DevPath   - Fv (6D99E806-3D38-42C2-A095-5F4300BFD7DC) /FuFile (462CAA21-7614-4503-B
36E-8AB6F4662331) /Enter Setup
Optional- N
FS0:\> _
```

Use the Dmpstore to Show the Boot Order

At the Shell Prompt:

FS0:> Dmpstore BootOrder

```
FS0:\> dmpstore bootorder
Variable NU+RT+BS 'EFIGlobalVariable:BootOrder' DataSize = 0x0C
  00000000: 05 00 01 00 02 00 03 00-04 00 00 00      *.....*
FS0:\> _
```

Use the BCFG to Move a boot item

Use BCFG to Move the 5th boot item
too 1st location.

Then verify using the “dmpstore”

(Hint: use BCFG -? -b for help menu)

The dmpstore output should look like
the screen shot



Result

```
FS0:\> dmpstore bootorder
Variable NU+RT+BS 'EFIGlobalVariable:BootOrder' DataSize = 0x0C
00000000: 00 00 05 00 01 00 02 00-03 00 04 00 *.....
```

Use the BCFG to Add a boot item

Copy the old EFI Shell from

`~/src/edk2-ws/edk2/ShellPkg/OldShell/Shell_FullX64.efi`

to the run-ovmf directory `~/run-ovmf/hda-contents`

Use BCFG to Add a 06 entry for a new boot option with `Shell_FullX64.efi`

Then verify using the “BCFG Boot Dump”

Hint: make sure `Shell_FullX64.efi` is in the `FS0:` directory by doing:

`FS0:\> Dir`

After the `bcfg add`, The output should look like

Exit QEMU



```
FS0:\> dir shell*.efi
Directory of: FS0:\

08/26/2021  15:33                771,136  Shell_FullX64.efi
```



Optional- Y	Result
Option: 06. Variable: Boot0006	
Desc - Olde EFI Shell 1.0	
DevPath - VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)/VenHw(6459-11D2-8E39-00A0C969723B,00000000)/\Shell_FullX64.efi	
Optional- N	
FS0:\> _	

SUMMARY

- ✱ Build a EDK II Platform using OVMF package
- ✱ Run Ovmf using Qemu



Questions?



Return to Main Training Page



Return to Training Table of contents for next presentation [link](#)



ACKNOWLEDGEMENTS

Redistribution and use in source (original document form) and 'compiled' forms (converted to PDF, epub, HTML and other formats) with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code (original document form) must retain the above copyright notice, this list of conditions and the following disclaimer as the first lines of this file unmodified.

Redistributions in compiled form (transformed to other DTDs, converted to PDF, epub, HTML and other formats) must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS DOCUMENTATION IS PROVIDED BY TIANOCORE PROJECT "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL TIANOCORE PROJECT BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS DOCUMENTATION, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Copyright (c) 2021-2022, Intel Corporation. All rights reserved.

Pre-requisites Ubuntu 16.04

Instructions from: [tianocore wiki Ubuntu 16.04](#)

Example Ubuntu 16.04

The following need to be accessible for building Edk2, From the terminal prompt (Cnt-Alt-T):

```
bash$ sudo apt-get install build-essential uuid-dev iasl git gcc-5 nasm python3-distutils
```

build-essential - Informational list of build-essential packages

uuid-dev - Universally Unique ID library (headers and static libraries)

iasl - Intel ASL compiler/decompiler (also provided by acpica-tools)

git - support for git revision control system

gcc-5 - GNU C compiler (v5.4.0 as of Ubuntu 16.04 LTS)

nasm - General-purpose x86 assembler

python3 - distutils - distutils module from the Python standard library

```
bash$ sudo apt-get install qemu
```

Qemu – Emulation with Intel architecture with UEFI Shell



ubuntu