

# UEFI & EDK II Training

UEFI Driver Wizard Lab

Currently only available in Ubuntu 16.04

[tianocore.org](http://tianocore.org)

# Lesson Objective

Linux Ubuntu 16.04 is only supported

Python Version 2.7 and  
python-wxgtk V3.0

Non-Ubuntu - Continue to [Porting UEFI Driver Lab](#)

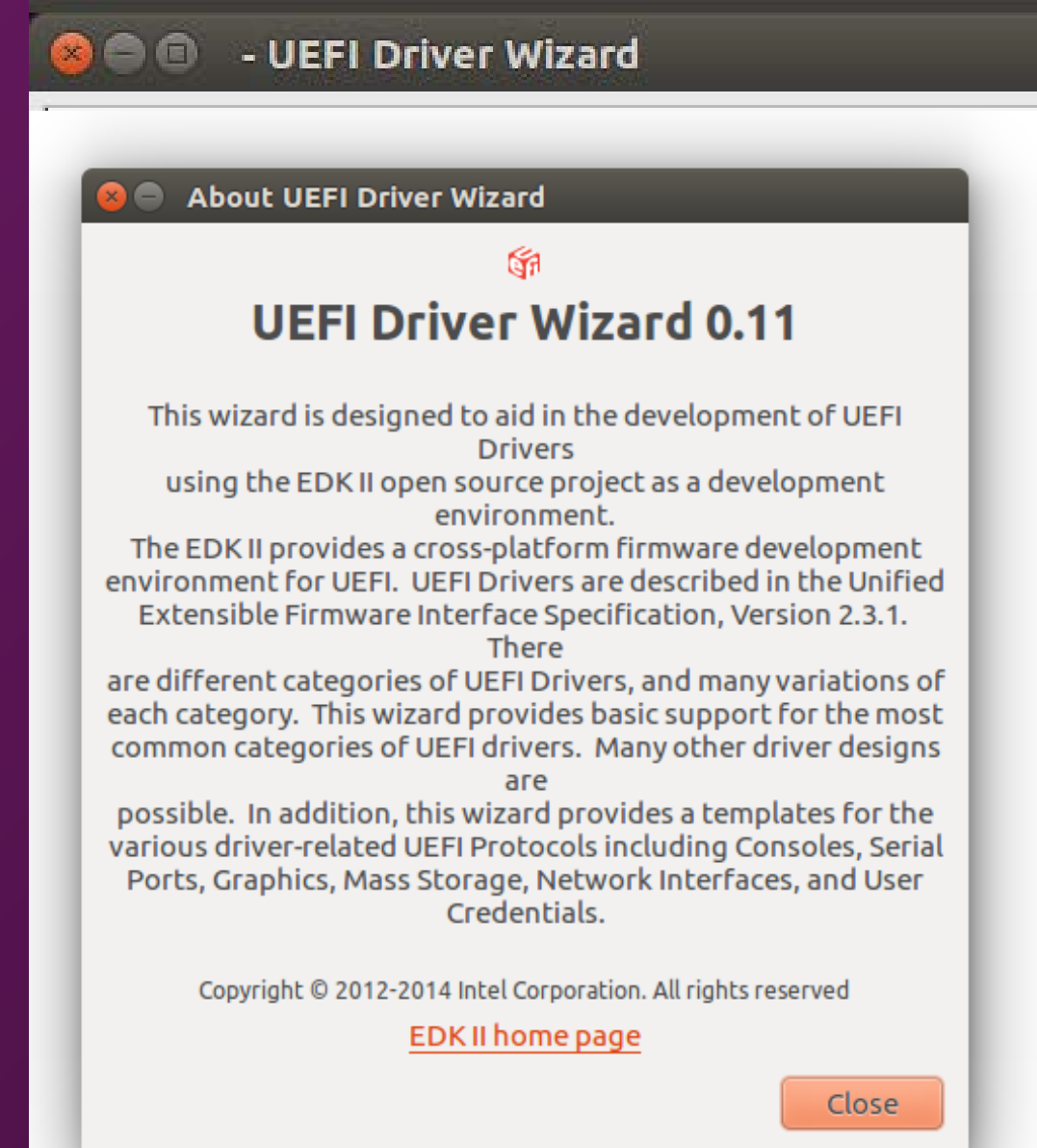
- ✿ Setup the UEFI Driver Wizard
- ✿ Create a UEFI Driver Template

# UEFI DRIVER WIZARD

Creating a Template UEFI Driver with the UEFI Driver Wizard

# UEFI Driver Wizard Overview

- ✓ Open source tool
- ✓ Based on *Driver Writer's Guide for UEFI 2.3.1* content
- ✓ Intel SSG engineers contributed
- ✓ Located on [www.TianoCore.org](http://www.TianoCore.org)



# Installing Python for UEFI Driver Wizard

## Requirements and Options

- Work space must contain BaseTools, MdePkg & MdeModulePkg Packages from [UDK2017](#) for Driver development on Tianocore.org
- Uses previous lab's setup \$HOME/fw/edk2-ws
- Python\* scripts from [GitHub Link](#) then use instructions from README for Python and wxPython versions to install then run

```
bash$ python launch.py
```

# Requirements for Your Driver



## Using UEFI Driver Wizard

- UEFI Device Driver
- UEFI Version 2.7 (0x00020046)  
`#define EFI_2_70_SYSTEM_TABLE_REVISION ((2<<16)|(70DEC))`
- Unloadable driver
- Support IA32 & x64 CPUs
- Returns component name information
- Byte stream device (i.e. UART / Serial I/O)
- Option to produce HII strings & forms for setup

# Template File Contents

Proper UEFI driver entry point

Basic driver libraries/headers

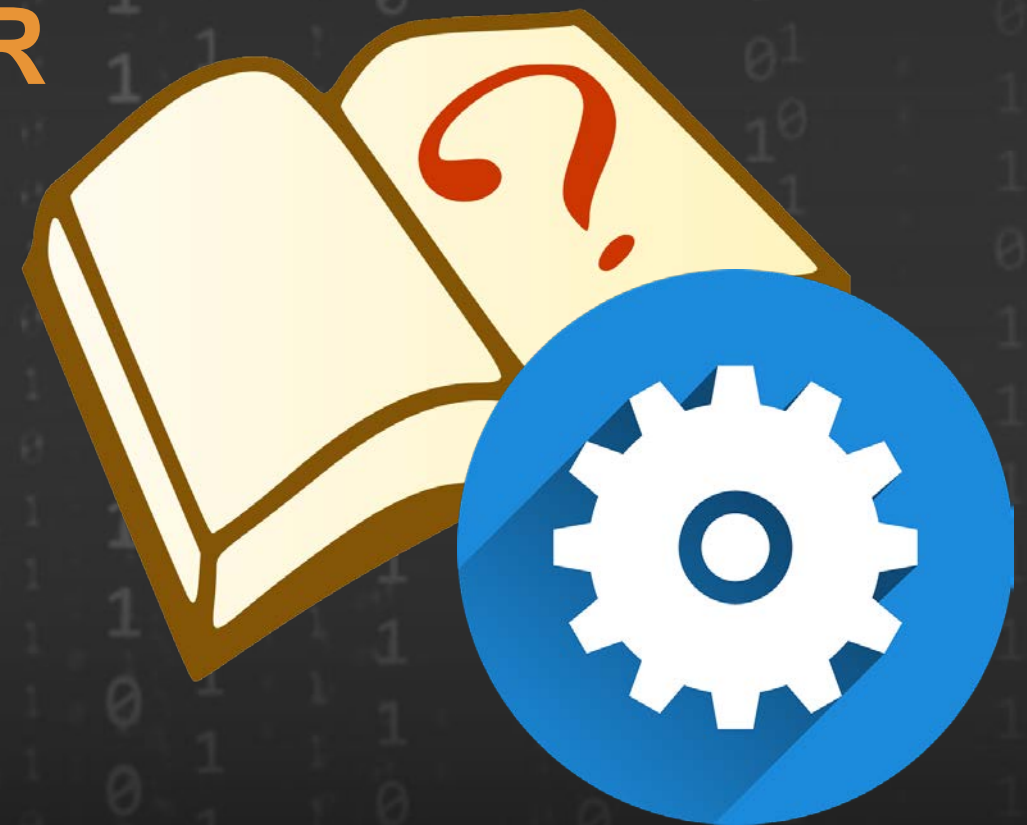
Skeletons for common driver functions

Error values until ported  
EFI\_UNSUPPORTED, EFI\_DEVICE\_ERROR



# LAB 1: CREATE A UEFI DRIVER WITH THE UEFI DRIVER WIZARD

- In this lab, you'll create a new UEFI driver using the UEFI Driver Wizard.
- This will create a set of "c" code files to be used as a template UEFI Driver used in the subsequent driver labs





# Lab 1: Install UEFI Driver Wizard, Python & wxPython

1. Perform Lab Setup from previous Labs
2. From the ~FW/DriverWizard folder, copy and paste folder “~FW/DriverWizard/UefiDriverWizard” to ~\$Home
3. Check if version 2.7.x is the default of Python from Terminal Prompt

```
bash$ python -V
```

```
Python 2.7.12
```

4. Install the wxPython (Version 3.0 )

```
bash$ sudo apt-get install python-wxgtk3.0
```

# Lab 1: UefiDriverWizard -Select Work Space

Terminal Prompt (Cnt-Alt-T)

```
bash$ cd ~UefiDriverWizard
bash$ python launch.py
```

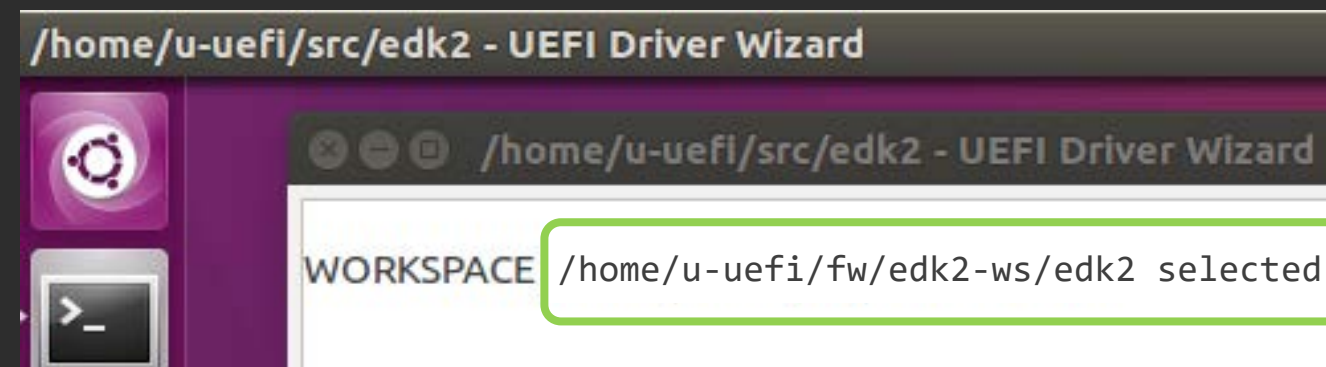
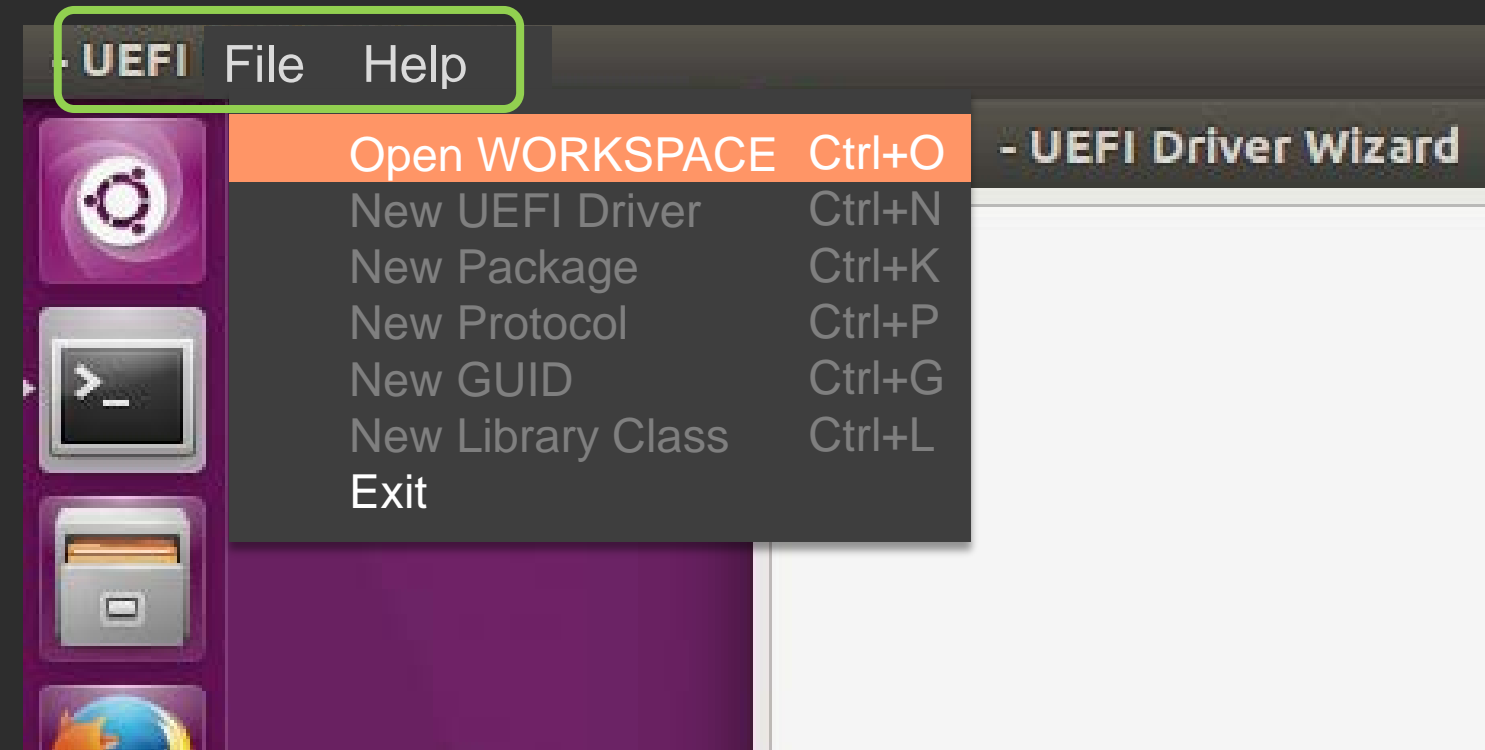
Select a Work Space

Control+O – to browse for a directory

Browse to ~/fw//edk2-ws/edk2

Select

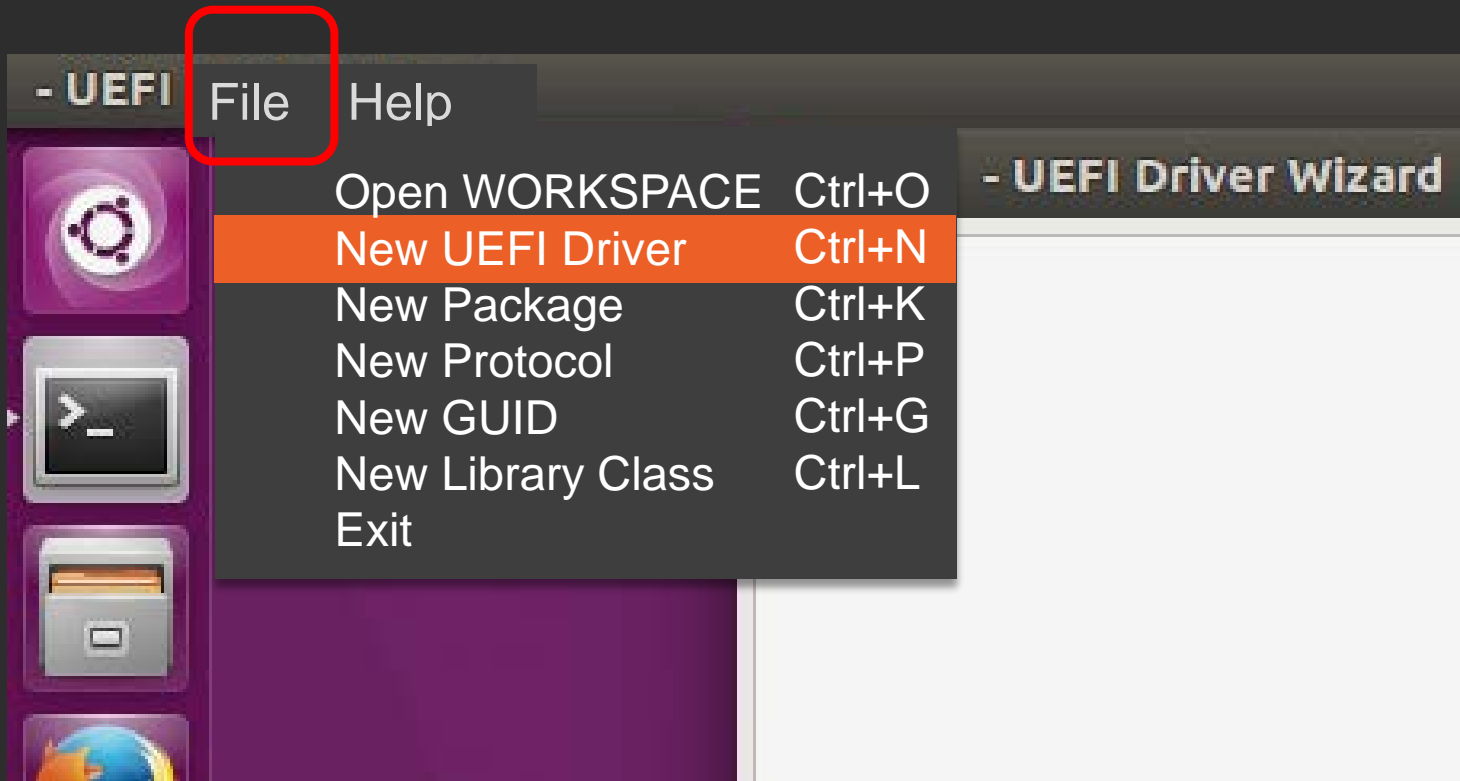
Open



Note: the environment for EDK II must be setup with edksetup.sh

# Lab 1: Create a New UEFI Driver

Control+N – to Open Menu



**New UEFI Driver**

UEFI Driver Path:

UEFI Driver Name:

UEFI Driver Version:

UEFI Driver GUID:

UEFI Driver Type:

- ☒ UEFI Driver Model Device Driver
- ☐ Root Bridge Driver
- ☐ UEFI Driver Model Bus Driver
- ☐ Service Driver
- ☐ UEFI Driver Model Hybrid Driver
- ☐ Initializing Driver

Driver Binding:

Optional Features Common to all UEFI Driver Types:

- ☒ Unloadable
- ☒ Driver Supported EFI Version Protocol
- ☒ HII Packages for Strings, Fonts, or Images

UEFI Specification Version:

CPU Architectures:

- ☐ All CPU Architectures
- ☒ IA32
- ☒ X64
- ☐ IPF
- ☐ EBC

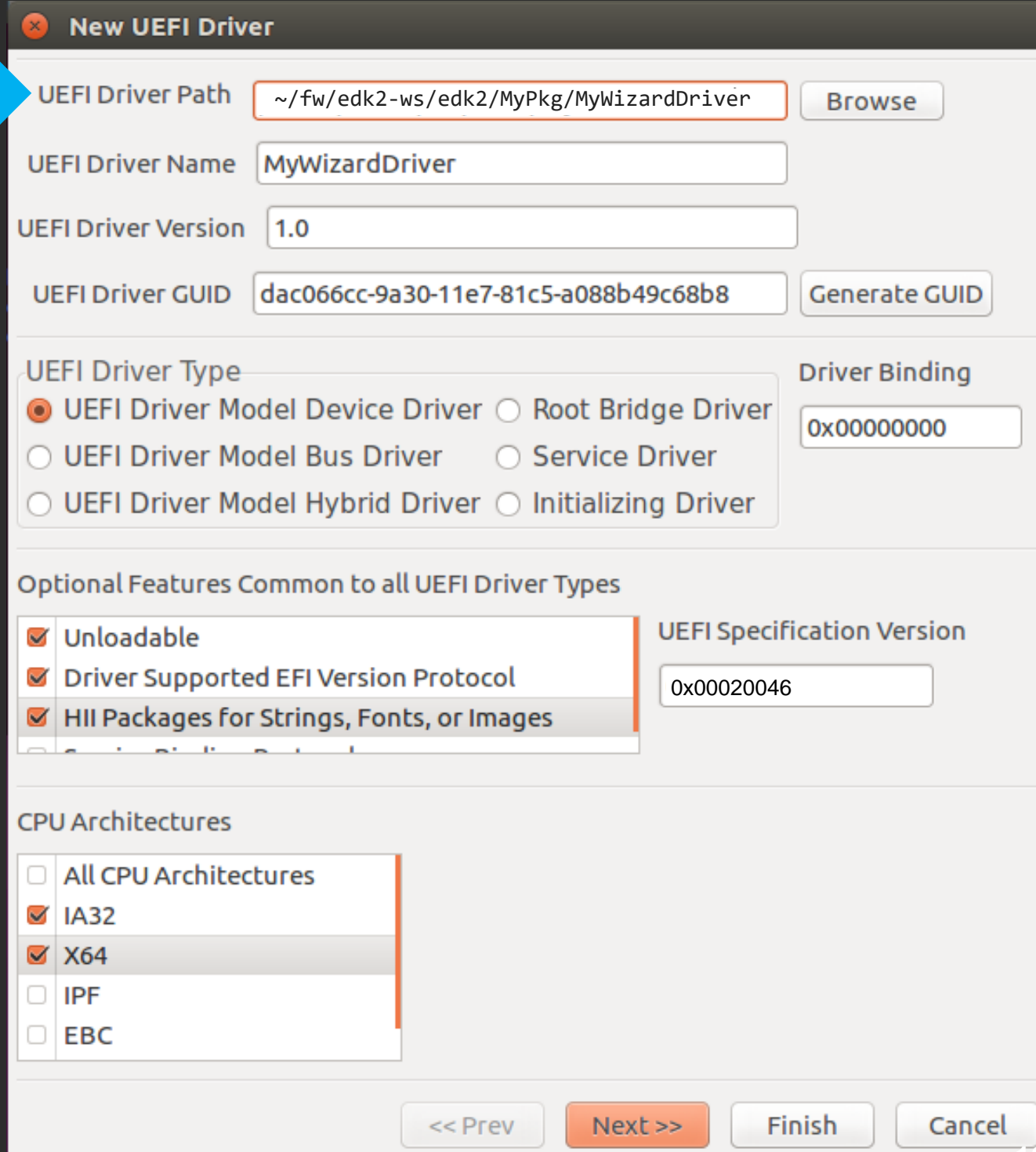
<< Prev **Next >>** Finish Cancel

# Lab 1: New UEFI Driver Menu

- UEFI Driver Path” – Type: “MyPkg/MyWizardDriver”  
*Note:* “UEFI Driver Name” is filled in.
- **Ensure** all the forms, radio buttons, and boxes are filled in and **selected** *exactly* like the image to the right.
- *Note:* A new, specific driver GUID will populate, so it will be different than this image

Click

Next >>



**New UEFI Driver**

UEFI Driver Path: ~/fw/edk2-ws/edk2/MyPkg/MyWizardDriver Browse

UEFI Driver Name: MyWizardDriver

UEFI Driver Version: 1.0

UEFI Driver GUID: dac066cc-9a30-11e7-81c5-a088b49c68b8 Generate GUID

UEFI Driver Type:

- ☒ UEFI Driver Model Device Driver
- ☐ Root Bridge Driver
- ☐ UEFI Driver Model Bus Driver
- ☐ Service Driver
- ☐ UEFI Driver Model Hybrid Driver
- ☐ Initializing Driver

Driver Binding: 0x00000000

Optional Features Common to all UEFI Driver Types:

- ☒ Unloadable
- ☒ Driver Supported EFI Version Protocol
- ☒ HII Packages for Strings, Fonts, or Images

UEFI Specification Version: 0x00020046

CPU Architectures:

- ☐ All CPU Architectures
- ☒ IA32
- ☒ X64
- ☐ IPF
- ☐ EBC

<< Prev **Next >>** Finish Cancel

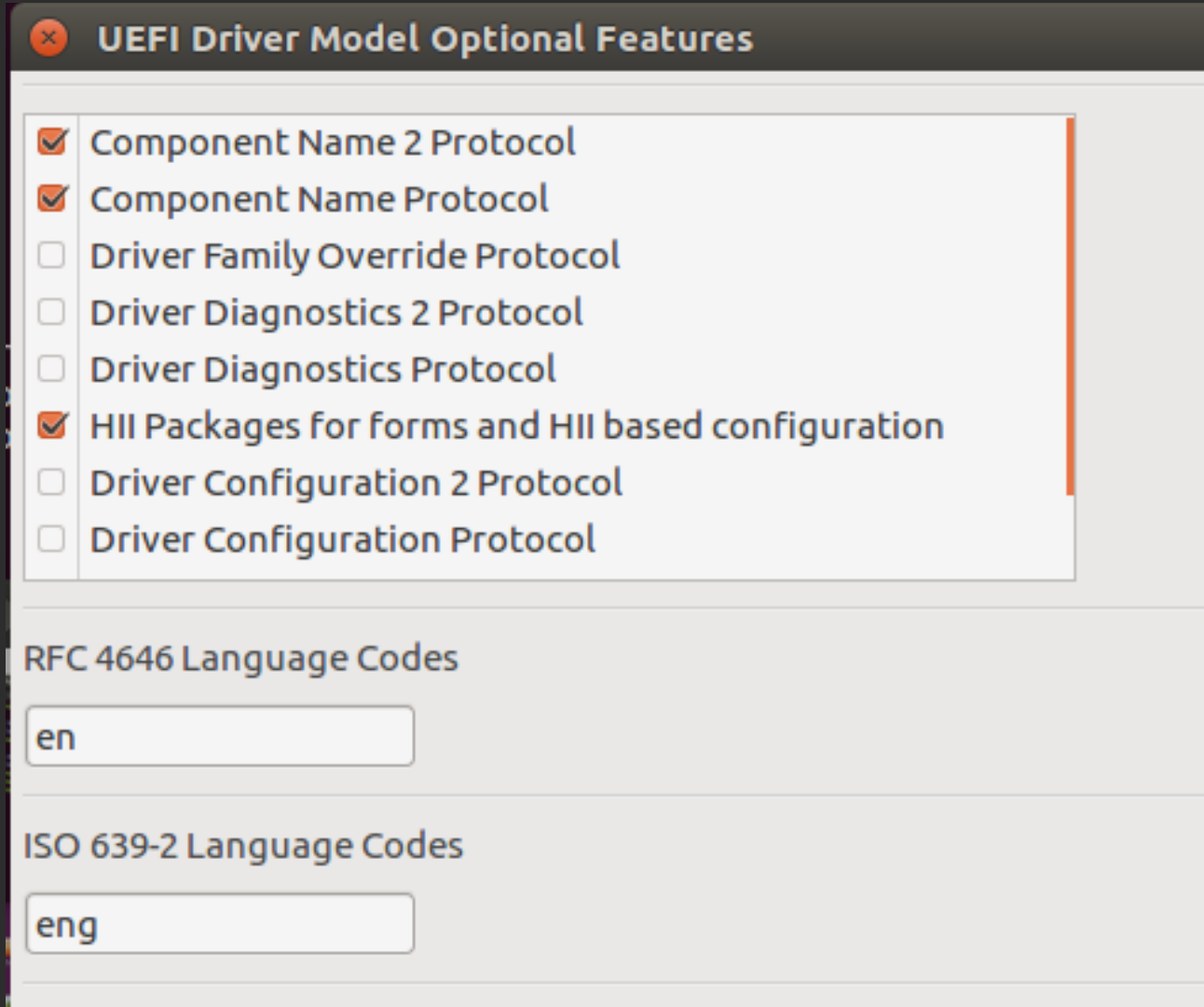
# Lab 1: UEFI Driver Model Optional Features

Ensure all the forms, radio buttons, and boxes are filled in and **selected exactly** like the image to the right.

- ✓ "Component Name 2 Protocol"
- ✓ "Component Name Protocol"
- ✓ "HII Packages for Forms . . ."

Click

Next >>



UEFI Driver Model Optional Features

- ☒ Component Name 2 Protocol
- ☒ Component Name Protocol
- ☐ Driver Family Override Protocol
- ☐ Driver Diagnostics 2 Protocol
- ☐ Driver Diagnostics Protocol
- ☒ HII Packages for forms and HII based configuration
- ☐ Driver Configuration 2 Protocol
- ☐ Driver Configuration Protocol

RFC 4646 Language Codes

en

ISO 639-2 Language Codes

eng

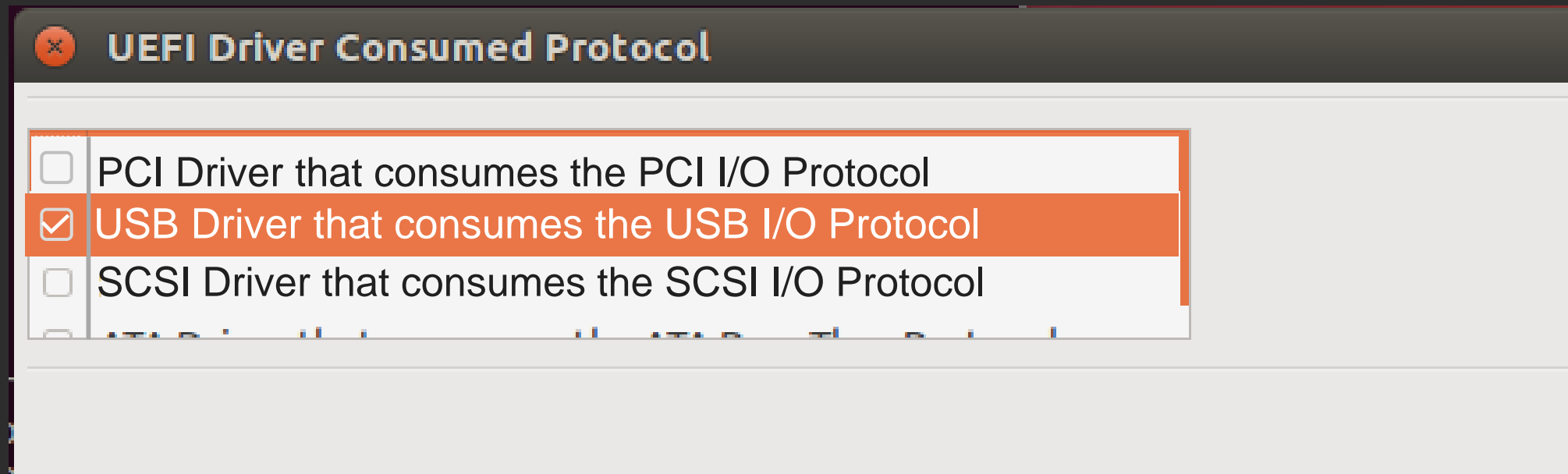
# Lab 1: UEFI Driver Consumed Protocol

Select

✓ “USB Driver that consumes the USB I/O Protocol”

Click

Next >>



The screenshot shows a window titled "UEFI Driver Consumed Protocol" with a close button in the top-left corner. Inside the window, there is a list of three options, each with a checkbox on the left:

- ☐ PCI Driver that consumes the PCI I/O Protocol
- ☒ USB Driver that consumes the USB I/O Protocol
- ☐ SCSI Driver that consumes the SCSI I/O Protocol

The second option, "USB Driver that consumes the USB I/O Protocol", is selected and highlighted with an orange background.



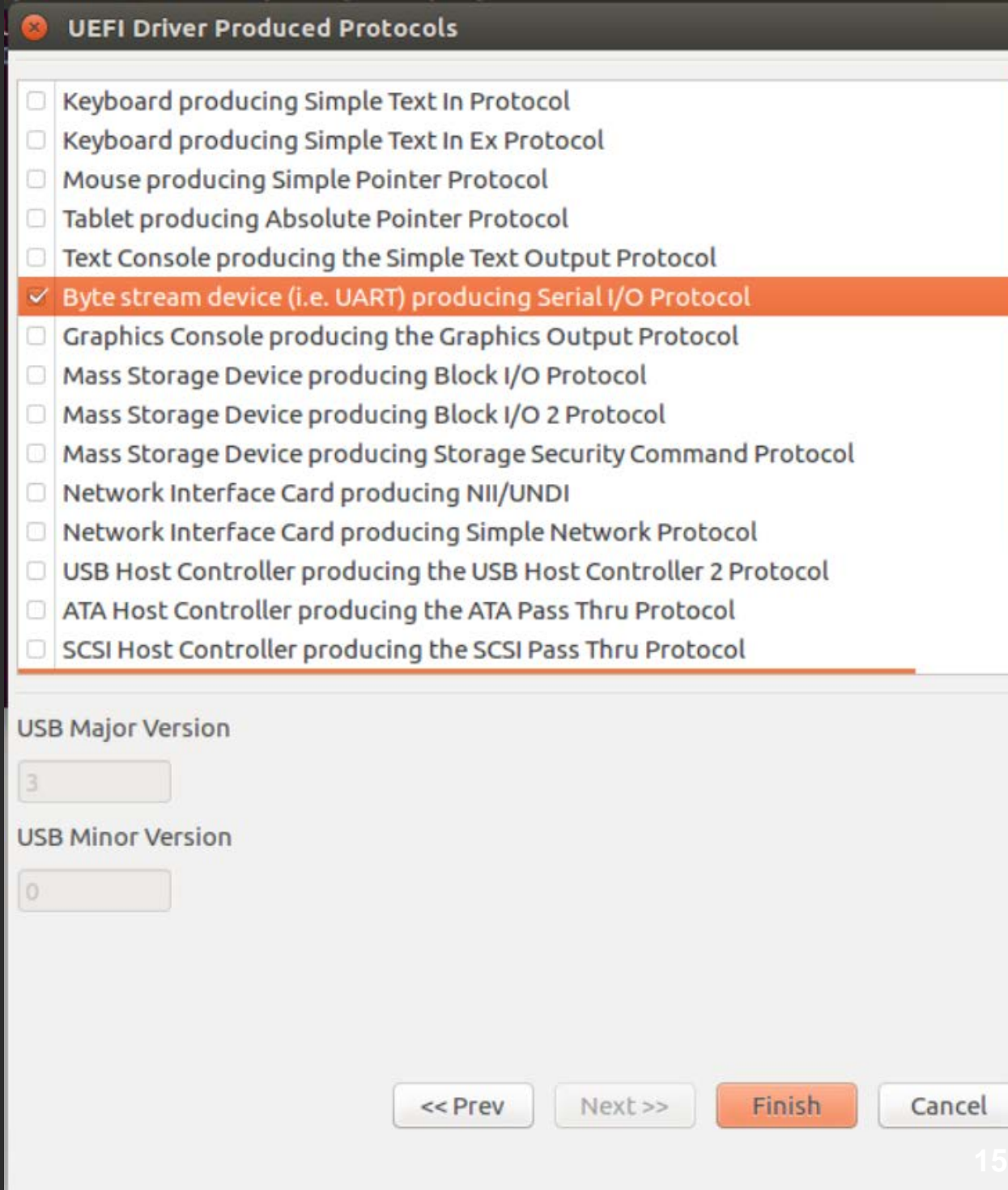
# Lab1: UEFI Driver Produced Protocols

## Select

- ✓ "Byte stream device (i.e. UART) producing Serial I/O Protocol"

Click

Finish



UEFI Driver Produced Protocols

- ☐ Keyboard producing Simple Text In Protocol
- ☐ Keyboard producing Simple Text In Ex Protocol
- ☐ Mouse producing Simple Pointer Protocol
- ☐ Tablet producing Absolute Pointer Protocol
- ☐ Text Console producing the Simple Text Output Protocol
- ☒ Byte stream device (i.e. UART) producing Serial I/O Protocol
- ☐ Graphics Console producing the Graphics Output Protocol
- ☐ Mass Storage Device producing Block I/O Protocol
- ☐ Mass Storage Device producing Block I/O 2 Protocol
- ☐ Mass Storage Device producing Storage Security Command Protocol
- ☐ Network Interface Card producing NII/UNDI
- ☐ Network Interface Card producing Simple Network Protocol
- ☐ USB Host Controller producing the USB Host Controller 2 Protocol
- ☐ ATA Host Controller producing the ATA Pass Thru Protocol
- ☐ SCSI Host Controller producing the SCSI Pass Thru Protocol

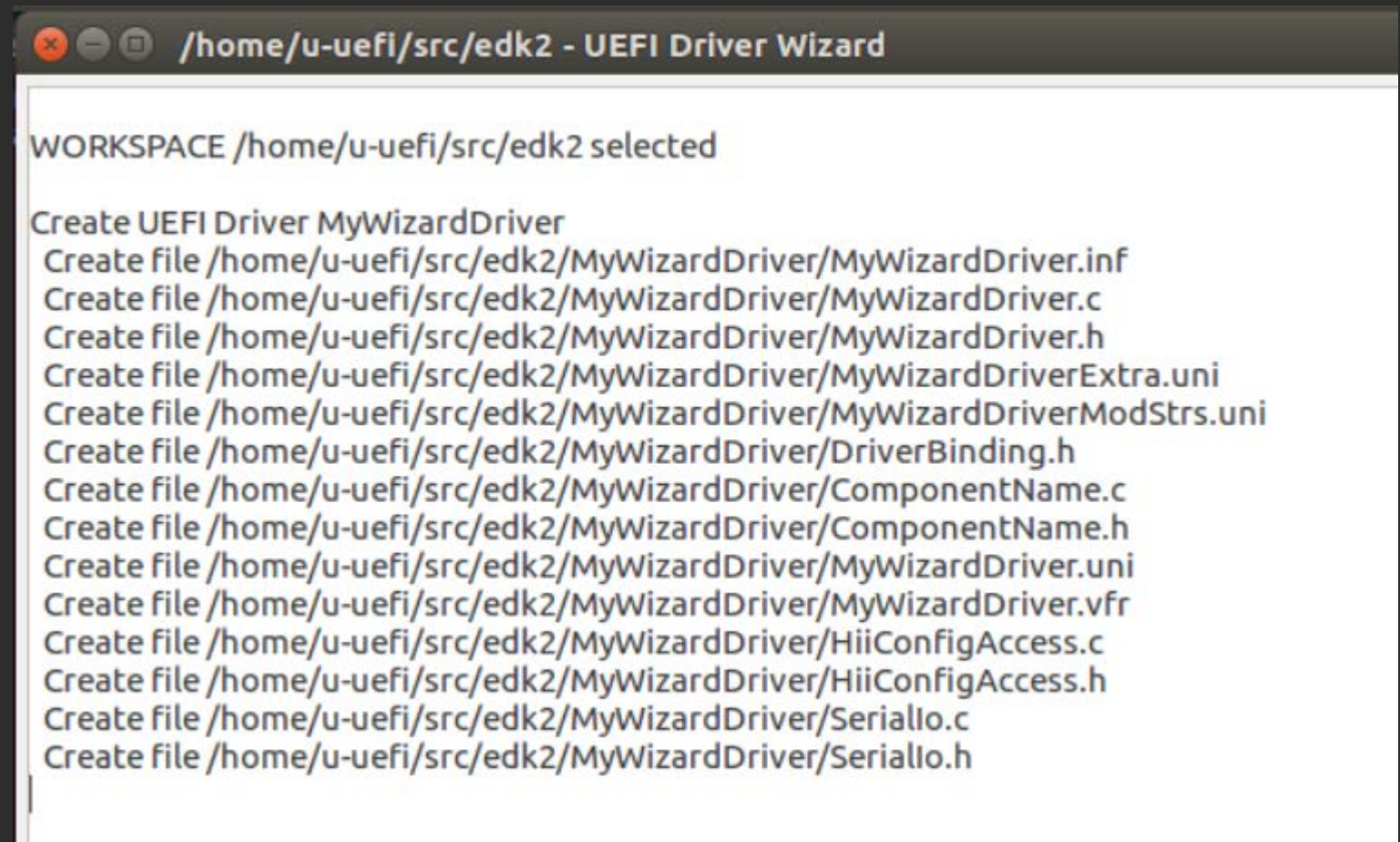
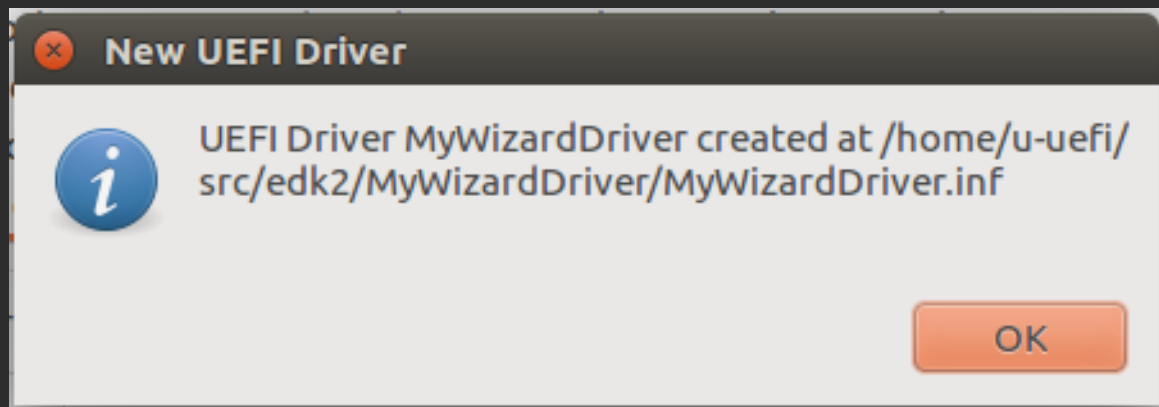
USB Major Version  
3

USB Minor Version  
0

<< Prev Next >> Finish Cancel

# Lab 1: UEFI Driver Created

UEFI Driver template created



Note: The New Driver will be created in the following:

`/home/u-uefi/fw/edk2-ws/edk2/MyPkg/MyWizardDriver`

# Summary

- ✿ Setup the UEFI Driver Wizard
- ✿ Create a UEFI Driver Template



# 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, Intel Corporation. All rights reserved.