tianocore

UEFI & EDK II TRAINING

UEFI Shell Lab – Ovmf with QEMU

tianocore.org

Lesson Objective

- Run UEFI Shell (QEMU)
- Run UEFI Shell Commands
- Run UEFI Shell Scripts

tianocore

UEFI Shell Lab with QEMU

QEMU Running OVMF

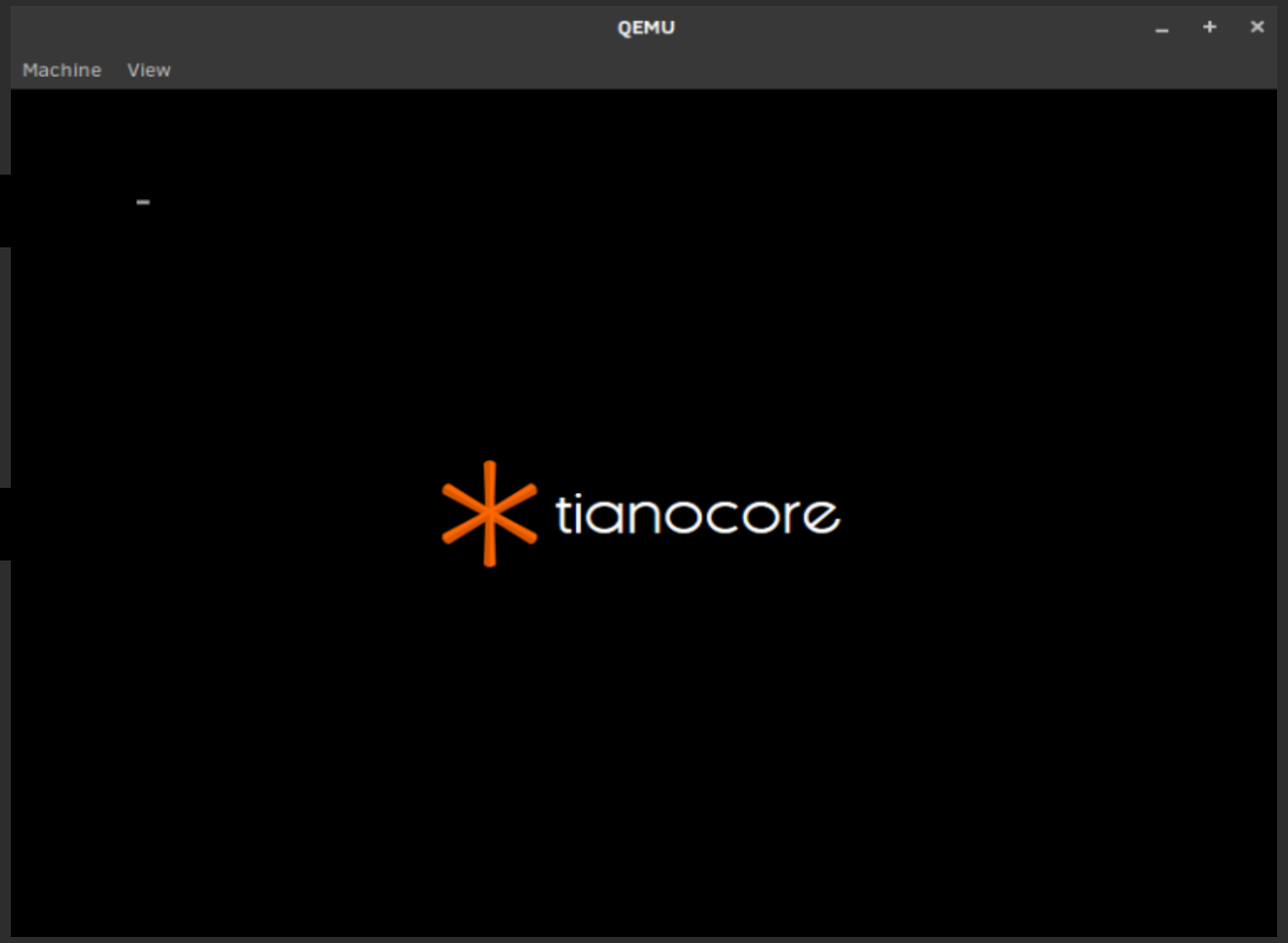
First Setup for Building EDK II for OVMF, See Lab Setup

1. Change to run-ovmf directory

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

2. Run the RunQemu.sh Linux shell script

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



See Platform Build lab for setup OVMF Lab Setup

QEMU Running OVMF

Type “F2” to get into the emulation setup or “Exit” from the Shell prompt

QEMU boot to UEFI Shell

QEMU

```
UEFI Interactive Shell v2.0. UEFI v2.31 (EDK II, 0x00010000). Revision 1.02
Mapping table
      FS0: Alias(s):HD8a1::BLK3:
          PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)/HD(1,MBR,0xBE1AFDFA,0x3F,0xFBFB9)
     BLK2: Alias(s):
          PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)
     BLK4: Alias(s):
          PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)
     BLK0: Alias(s):
          PciRoot(0x0)/Pci(0x1,0x0)/Floppy(0x0)
     BLK1: Alias(s):
          PciRoot(0x0)/Pci(0x1,0x0)/Floppy(0x1)
Press ESC in 0 seconds to skip startup.nsh or any other key to continue.
Shell> _
```

tianocore

UEFI SHELL COMMANDS

Commands from the Command Line Interface

COMMON SHELL COMMANDS FOR DEBUGGING

```
help
mm
mem
memmap
drivers
devices
devtree
dh
Load
Dmpstore
pci
stall
```

"**-b**" is the command line parameter for breaking after each page.

Shell Help

```
Shell> help -b
```

QEMU

```
alias         - Displays, creates, or deletes UEFI Shell aliases.
attrib        - Displays or modifies the attributes of files or directories.
bcfg          - Manages the boot and driver options that are stored in NVRAM.
cd            - Displays or changes the current directory.
cls           - Clears the console output and optionally changes the background
and foreground color.
comp          - Compares the contents of two files on a byte-for-byte basis.
connect       - Binds a driver to a specific device and starts the driver.
cp            - Copies one or more files or directories to another location.
date          - Displays and sets the current date for the system.
dblk          - Displays one or more blocks from a block device.
devices       - Displays the list of devices managed by UEFI drivers.
devtree       - Displays the UEFI Driver Model compliant device tree.
dh            - Displays the device handles in the UEFI environment.
disconnect    - Disconnects one or more drivers from the specified devices.
dmem          - Displays the contents of system or device memory.
dmpstore      - Manages all UEFI variables.
drivers       - Displays the UEFI driver list.
drvcfg        - Invokes the driver configuration.
drvdiag       - Invokes the Driver Diagnostics Protocol.
echo          - Controls script file command echoing or displays a message.
edit          - Provides a full screen text editor for ASCII or UCS-2 files.
eficompress   - Compresses a file using UEFI Compression Algorithm.
efidecompress - Decompresses a file using UEFI Decompression Algorithm.
Press ENTER to continue or 'Q' break:_
```

Shell “memmap”

```
Shell> memmap
```

Displays the memory map maintained by the UEFI environment

```
Available 00000000061C0000-000000000A1BFFFF 0000000000004000 000000000000000F
MMIO      00000000021A0000-00000000021ABFFF 000000000000000C 8000000000000000
  Reserved  :          4 Pages (16,384)
  LoaderCode:        358 Pages (1,466,368)
  LoaderData:         23 Pages (94,208)
  BS_Code   :        550 Pages (2,252,800)
  BS_Data   :      3,895 Pages (15,953,920)
  RT_Code   :         64 Pages (262,144)
  RT_Data   :         64 Pages (262,144)
  ACPI Recl :          0 Pages (0)
  ACPI NVS  :          0 Pages (0)
  MMIO      :         12 Pages (49,152)
  Available :     27,810 Pages (113,909,760)
Total Memory: 128 MB (134,266,880 Bytes)
Shell> _
```

Shell “mm”

```
Shell> mm -? -b
```

Help for “mm” command shows options for different types of memory and I/O that can be modified

QEMU

```
Displays or modifies MEM/MMIO/IO/PCI/PCIE address space.

MM Address [Value] [-w 1|2|4|8] [-MEM | -MMIO | -IO | -PCI | -PCIE] [-n]

    Address  - Starting address
    Value    - The value to write
    -MEM     - Memory Address type
    -MMIO    - Memory Mapped IO Address type
    -IO      - IO Address type
    -PCI     - PCI Configuration Space Address type:
               Address format: 0x000000ssbbddffrr
                 ss   - Segment
                 bb   - Bus
                 dd   - Device
                 ff   - Function
                 rr   - Register
    -PCIE    - PCIE Configuration Space Address type:
               Address format: 0x00000ssbbddffrrr
                 ss   - Segment
                 bb   - Bus
                 dd   - Device
                 ff   - Function
                 rrr  - Register
    -w       - Unit size accessed in bytes:
Press ENTER to continue, 'q' to exit:_
```

Shell “mm”

** `Shell> mm 06bbb000`

```
Shell> mm 06bbb000
MEM  0x0000000006BBB000 : 0xAF >

MEM  0x0000000006BBB001 : 0xAF >

MEM  0x0000000006BBB002 : 0xAF >

MEM  0x0000000006BBB003 : 0xAF >

MEM  0x0000000006BBB004 : 0xAF >

MEM  0x0000000006BBB005 : 0xAF >

MEM  0x0000000006BBB006 : 0xAF >

MEM  0x0000000006BBB007 : 0xAF > q

Shell> _
```

**Pick a location from the `MemMap` command on Previous slide

`MM` in can display / modify any location

Try

```
Shell> mm 0000
```

“q” to quit

Shell “mem”

```
Shell> mem
```

Displays the contents of the system or device memory

without arguments, displays the system memory configuration.

```
  061EC170: AF AF AF AF AF AF AF AF-AF AF AF AF AF AF AF AF  *................*
  061EC180: AF AF AF AF AF AF AF AF-AF AF AF AF AF AF AF AF  *................*

Valid EFI Header at Address 00000000061EBF90
-------------------------------------------
System: Table Structure size 00000048 revision 0002001F
ConIn (000000000A3271F4) ConOut (0000000005373114) StdErr (000000000A3273A4)
Runtime Services 00000000061EBF10
Boot Services    0000000000415C40
SAL System Table 0000000000000000
ACPI Table       0000000000000000
ACPI 2.0 Table   0000000000000000
MPS Table        0000000000000000
SMBIOS Table     000000000622F000
Shell> _
```

UEFI System Table Pointer

Shell “Drivers”

```
Shell> drivers -b
```

```
           T   D
D          Y C I
R          P F A
V  VERSION E G G #D #C DRIVER NAME                         IMAGE NAME
=== ======= = = = == == ================================== ==========
 42 0000000A B N N 1  6                      PCI Bus Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(93B80004-9FB3-11D4-9A3A-0090273FC14D)
 44 00000011 ? N N 0  0        Block MMIO to Block IO Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(33CB97AF-6C33-4C42-986B-07581FA366D4)
 45 00000010 ? N N 0  0                 Virtio Block Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(11D92DFB-3CA9-4F93-BA2E-4780ED3E03B5)
 46 00000010 ? N N 0  0              Virtio SCSI Host Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(FAB5D4F4-83C0-4AAF-8480-442D11DF6CEA)
 47 0000000A D N N 2  0   Platform Console Management Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(51CCF399-4FDF-4E55-A45B-E123F84D456A)
 48 0000000A D N N 2  0   Platform Console Management Driver
 49 0000000A B N N 2  2              Console Splitter Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(408EDCEC-CF6D-477C-A5A8-B4844E3DE281)
 4A 0000000A ? N N 0  0              Console Splitter Driver
 4B 0000000A ? N N 0  0              Console Splitter Driver
 4C 0000000A B N N 2  2              Console Splitter Driver
 4D 0000000A B N N 1  1              Console Splitter Driver
 51 0000000A D N N 1  0              Graphics Console Driver MemoryMapped(0xB,0x
17A8E000,0x17FBDFFF)/FvFile(CCCB0C28-4B24-11D5-9A5A-0090273FC14D)
Press ENTER to continue or 'Q' break:_
```

Shell “Devices”

```
Shell> devices -b
```

Displays a list of devices that UEFI drivers manage.

```
Shell> devices -b
  C  T    D
  T  Y C  I
  R  P F  A
  L  E G  G #P #D #C Device Name
  == = = = == == == ==============================================
 32 R - -  0  1  6 PciRoot(0x0)
 4E D - -  2  0  0 Primary Console Input Device
 4F D - -  2  0  0 Primary Console Output Device
 50 D - -  1  0  0 Primary Standard Error Device
 7A D - -  1  0  0 PciRoot(0x0)/Pci(0x0,0x0)
 7B B - -  1  2  6 PciRoot(0x0)/Pci(0x1,0x0)
 7C B - X  1  2  2 PCI IDE/ATAPI Controller
 7D D - -  1  0  0 PciRoot(0x0)/Pci(0x1,0x3)
 7E D - -  1  1  0 QEMU Video PCI Adapter
 7F D - -  1  0  0 PciRoot(0x0)/Pci(0x3,0x0)
 80 B - -  1  2  1 QEMU HARDDISK
 81 D - -  1  1  0 QEMU DVD-ROM
 82 D - -  1  2  0 FAT File System
 83 R - -  0  3  1 PciRoot(0x0)/Pci(0x2,0x0)/AcpiAdr(0x80010100)
 84 B - -  1  1  1 PciRoot(0x0)/Pci(0x1,0x0)/Serial(0x0)
 85 D - -  1  0  0 PciRoot(0x0)/Pci(0x1,0x0)/Serial(0x1)
 86 B - -  1  3  1 PS/2 Keyboard Device
 87 D - -  1  0  0 PciRoot(0x0)/Pci(0x1,0x0)/Acpi(PNP0303,0x1)
Press ENTER to continue or 'Q' break:_
```

Shell “Devtree”

```
Shell> devtree -b
```

Displays s tree of devices currently managed by UEFI drivers.

```
 Ctrl[03] MemoryMapped(0xB,0x800000,0xFFFFFF)
 Ctrl[04] MemoryMapped(0xB,0x17A8E000,0x17FBDFFF)
 Ctrl[1B] MemoryMapped(0xB,0x17FE0000,0x17FFFFFF)
 Ctrl[32] PciRoot(0x0)
   Ctrl[7A] PciRoot(0x0)/Pci(0x0,0x0)
   Ctrl[7B] PciRoot(0x0)/Pci(0x1,0x0)
     Ctrl[84] PciRoot(0x0)/Pci(0x1,0x0)/Serial(0x0)
       Ctrl[8A] PciRoot(0x0)/Pci(0x1,0x0)/Serial(0x0)/Uart(115200,8,N,1)
         Ctrl[8B] PC-ANSI Serial Console
           Ctrl[4E] Primary Console Input Device
           Ctrl[4F] Primary Console Output Device
           Ctrl[50] Primary Standard Error Device
     Ctrl[85] PciRoot(0x0)/Pci(0x1,0x0)/Serial(0x1)
     Ctrl[86] PS/2 Keyboard Device
       Ctrl[4E] Primary Console Input Device
     Ctrl[87] PciRoot(0x0)/Pci(0x1,0x0)/Acpi(PNP0303,0x1)
     Ctrl[88] ISA Floppy Drive #0
     Ctrl[89] ISA Floppy Drive #1
   Ctrl[7C] PCI IDE/ATAPI Controller
     Ctrl[80] QEMU HARDDISK
       Ctrl[82] FAT File System
     Ctrl[81] QEMU DVD-ROM
   Ctrl[7D] PciRoot(0x0)/Pci(0x1,0x3)
   Ctrl[7E] QEMU Video PCI Adapter
Press ENTER to continue or 'Q' break:_
```

Shell Handle Database - “Dh”

```
Shell> dh -b
```

Displays the device handles associated with UEFI drivers

```
01: LoadedImage
02: Decompress
03: UnknownDevice DevicePath(yMapped(0xB,0x800000,0xFFFFFF))
 UnknownDevice
04: UnknownDevice DevicePath(ped(0xB,0x17A8E000,0x17FBDFFF))
 UnknownDevice
05: UnknownDevice
06: ImageDevicePath LoadedImage
07: UnknownDevice Pcd
08: ImageDevicePath LoadedImage
09: UnknownDevice
0A: ImageDevicePath LoadedImage
0B: UnknownDevice
0C: ImageDevicePath LoadedImage
0D: UnknownDevice UnknownDevice
0E: DebugSupport EBCInterpreter ImageDevicePath LoadedImage
0F: UnknownDevice
10: ImageDevicePath LoadedImage
11: UnknownDevice
12: ImageDevicePath LoadedImage
13: UnknownDevice
14: ImageDevicePath LoadedImage
15: UnknownDevice
16: ImageDevicePath LoadedImage
Press ENTER to continue or 'Q' break:_
```

Shell “Load”

```
Shell> load -?
```

Loads a UEFI driver into memory

```
Shell> load -?
Loads a UEFI driver into memory.
load [-nc] file [file...]
-nc   Load the driver, but do not connect the driver.
File  File that contains the image of the UEFI driver (wildcards are permitted)
This command loads an driver into memory. It can load multiple files at one time
, and the file name supports wildcards.
If the -nc flag is not specified, this command will try to connect the driver to
 a proper device; it may also cause already loaded drivers be connected to their
 corresponding devices.
fs0:\> load Isabus.efi
load: Image 'fs0:\Isabus.efi' loaded at 18FE000 - Success
fs0:\> load Isabus.efi IsaSerial.efi
load: Image 'fs0:\Isabus.efi' loaded at 18E5000 - Success
load: Image 'fs0:\IsaSerial.efi' loaded at 18DC000 - Success
fs0:\> load Isa*.efi
load: Image 'fs0:\IsaBus.efi' loaded at 18D4000 - Success
load: Image 'fs0:\IsaSerial.efi' loaded at 18CB000 - Success
fs0:\> load -nc IsaBus.efi
load: Image 'fs0:\Isabus.efi' loaded at 18FE000 - Success
Shell> _
```

Shell “dmpstore”

```
Shell> dmpstore –all -b
```

Display the contents of the NVRAM variables

```
Shell> dmpstore -all -b
Variable NV+BS '4C19049F-4137-4DD3-9C10-8B97A83FFDFA:MemoryTypeInformation' Data
Size = 0x40
  00000000: 0A 00 00 00 2A 00 00 00-09 00 00 00 08 00 00 00  *....*...........*
  00000010: 00 00 00 00 29 00 00 00-06 00 00 00 F2 00 00 00  *....)...........*
  00000020: 05 00 00 00 50 00 00 00-03 00 00 00 57 03 00 00  *....P.......W...*
  00000030: 04 00 00 00 AC 14 00 00-0F 00 00 00 00 00 00 00  *................*
Variable NV+RT+BS 'EFIGlobalVariable:ErrOut' DataSize = 0x49
  00000000: 02 01 0C 00 D0 41 03 0A-00 00 00 00 01 01 06 00  *.....A..........*
  00000010: 00 01 02 01 0C 00 D0 41-01 05 00 00 00 00 03 0E  *.......A........*
  00000020: 13 00 00 00 00 00 00 C2-01 00 00 00 00 00 08 01  *................*
  00000030: 01 03 0A 14 00 53 47 C1-E0 BE F9 D2 11 9A 0C 00  *.....SG.........*
  00000040: 90 27 3F C1 4D 7F FF 04-00                       *.'?.M....*
Variable NV+RT+BS 'EFIGlobalVariable:ConIn' DataSize = 0x7A
  00000000: 02 01 0C 00 D0 41 03 0A-00 00 00 00 01 01 06 00  *.....A..........*
  00000010: 00 01 02 01 0C 00 D0 41-03 03 00 00 00 00 7F 01  *.......A........*P
ress ENTER to continue or 'Q' break:_
```

Shell “pci”

```
Shell> pci -? -b
```

Display the help for the PCI command

```
Shell> pci -? -b
Displays PCI device list or PCI function configuration space and PCIe extended
configuration space.

PCI [Bus Dev [Func] [-s Seg] [-i [-ec ID]]]

  -s   - Specifies optional segment number (hexadecimal number).
  -i   - Displays interpreted information.
  -ec  - Displays detailed interpretation of specified PCIe extended capability
         ID (hexadecimal number).
  Bus  - Specifies a bus number (hexadecimal number).
  Dev  - Specifies a device number (hexadecimal number).
  Func - Specifies a function number (hexadecimal number).

NOTES:
  1. This command displays a list of all the PCI devices found in the system. It

     also displays the configuration space of a PCI device according to the
     specified bus (Bus), device (Dev), and function (Func) addresses. If the
     function address is not specified, it defaults to 0.
  2. The -i option displays verbose information for the specified PCI
     device. The PCI configuration space for the device is displayed with
     a detailed interpretation.
  3. If no parameters are specified, all PCI devices are listed.
Press ENTER to continue or 'Q' break:_
```

Shell “Stall”

```
Shell> stall 10000000
```

Stalls the operation for a specified number of microseconds

```
Shell> stall 10000000
Shell> _
```

tianocore

UEFI SHELL SCRIPTS

Use Scripting with UEFI Shell

UEFI SHELL SCRIPTS

The UEFI Shell can execute commands from a file, which is called a batch script file (**.nsh** files).

Benefits: These files allow users to simplify routine or repetitive tasks.

- Perform basic flow control.
- Allow branching and looping in a script.
- Allow users to control input and output and call other batch programs (known as script nesting).

Writing UEFI Shell Scripts

At the shell prompt

```
Shell> fs0:
FS0:\> edit HelloScript.nsh
```

Type : `echo Hello World`

```
UEFI EDIT 2.0 HelloScript.nsh                    Modified
echo "Hello World"
```

Press “F2”
Enter
Press “F3” to exit

Help Menu - Shell

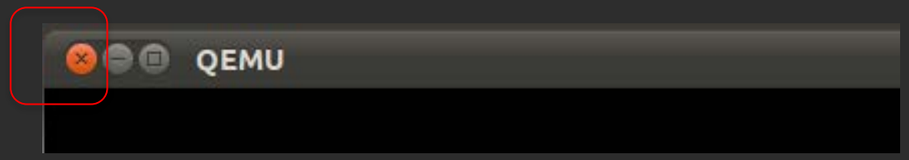
```
Help

Control Key   Function Key   Command
-----------   ------------   -----------------
Ctrl-G        F1             Go To Line
Ctrl-S        F2             Save File
Ctrl-Q        F3             Exit
Ctrl-F        F4             Search
Ctrl-R        F5             Search/Replace
Ctrl-K        F6             Cut Line
Ctrl-U        F7             Paste Line
Ctrl-O        F8             Open File
Ctrl-T        F9             File Type

Use Ctrl-W to exit this help
```

Hello World Script

In the shell, **type** HelloScript for the following result:

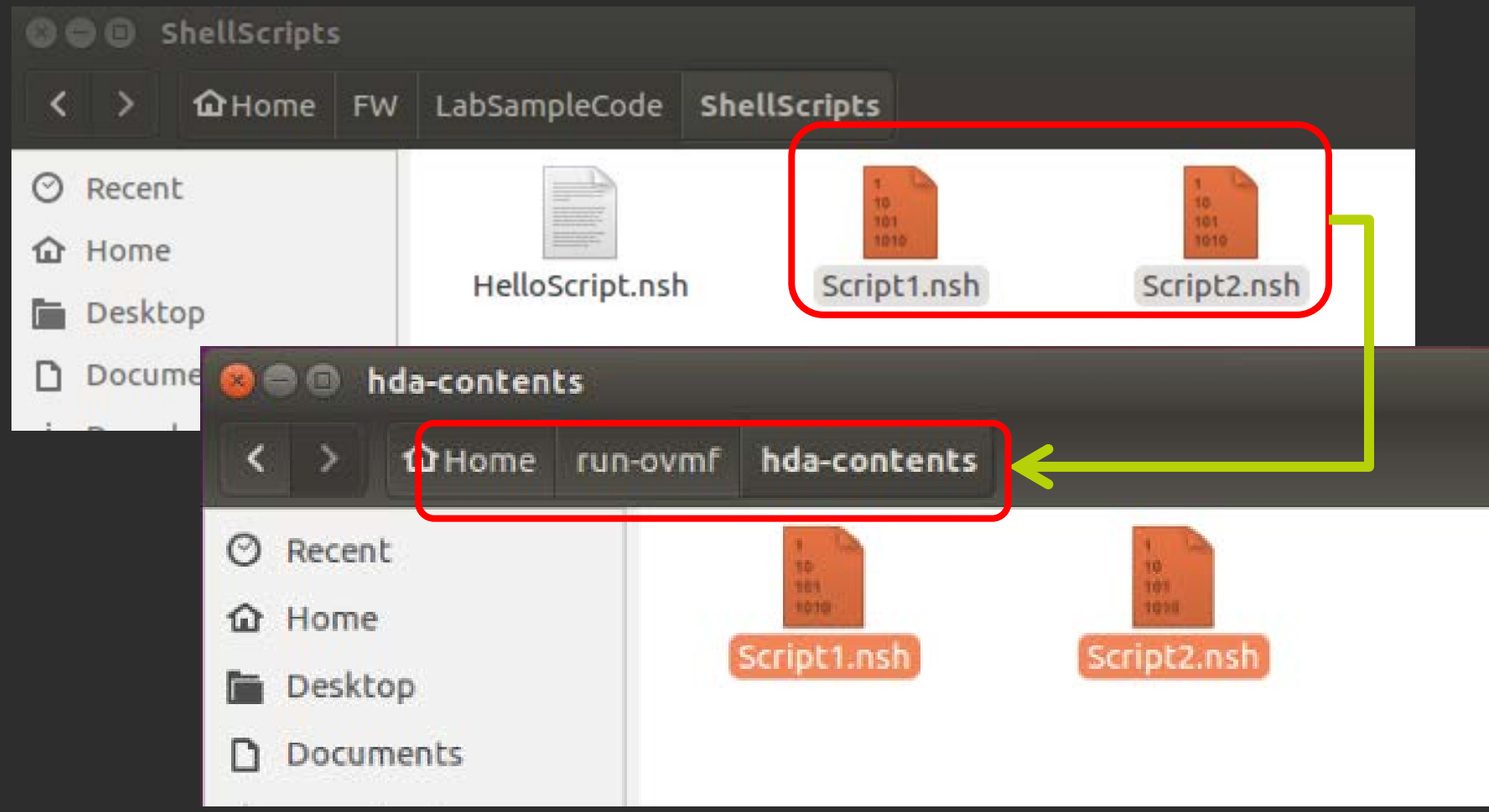
```
FS0:\> HelloScript.nsh
FS0:\> echo "Hello World"
Hello World
FS0:\> _
```

Close the QEMU



UEFI SHELL NESTED SCRIPTS

QEMU: Copy the Scripts from the `~/FW/LabSampleCode/ShellScripts` to the run-ovmf directory `~/run-ovmf/hda-contents`



UEFI Shell Script Example

Script1.nsh

```
# Simple UEFI Shell script file
echo -off
script2.nsh
if exist %cwd%Mytime.log then
     type Mytime.log
endif
echo "%HThank you.” “%VByeBye:) %N“
```

Script1.nsh

```
# Show nested scripts
time > Mytime.log
for %a run (3 1 -1)
    echo %a counting down
endfor
```

Run UEFI Shell Scripts

Run the RunQemu.sh from the terminal (Cnt-Alt-T)

```
bash$ cd ~run-ovmf
bash$ . RunQemu.sh
```

At the Shell prompt Type

```
Shell> fs0:
FS0:\> Script1

FS0:\> Edit Script1.nsh
```

```
FS0:\> Script1
FS0:\> script2.nsh
FS0:\> time > Mytime.log
FS0:\> for %a run (3 1 -1)
FS0:\>     echo %a counting down
3 counting down
FS0:\> endfor
FS0:\> for %a run (3 1 -1)
FS0:\>     echo %a counting down
2 counting down
FS0:\> endfor
FS0:\> for %a run (3 1 -1)
FS0:\>     echo %a counting down
1 counting down
FS0:\> endfor
FS0:\> for %a run (3 1 -1)
FS0:\> if exist %cwd%Mytime.log then
FS0:\>      type Mytime.log
20:08:54 (UTC 00:00)

FS0:\> endif
FS0:\> echo "Thank you. ByeBye:) "
Thank you. ByeBye:)
FS0:\> _
```

Run UEFI Shell Scripts

Remove the “#” on the first line

```
UEFI EDIT Script1.nsh
echo  -off
script2.nsh
if exist %cwd%Mytime.log then
     type Mytime.log
endif
echo "%HThank you. %VByeBye:) %N"
```

Press “F2”

Enter

Press “F3” to exit

Type

```
FS0:\> Script1
```

```
FS0:\> Script1
FS0:\> echo  -off
3 counting down
2 counting down
1 counting down
20:19:52 (UTC 00:00)

Thank you. ByeBye:)
FS0:\> _
```

tianocore

UEFI SHELL GLOBAL VARIABLES

Use BCFG and DmpStore

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)/FvFile(EEC25BDC-67F2-4D95-B
1D5-F81B2039D11D)
  Optional- N
Option: 02. Variable: Boot0002
  Desc    - UEFI Misc Device
  DevPath - VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)/VenHw(6888A4AE-
AFCE-E84B-9102-F7B9DAE6A030,00000000)
  Optional- Y
Option: 03. Variable: Boot0003
  Desc    - UEFI Non-Block Boot Device
  DevPath - VenHw(5CF32E0B-8EDF-2E44-9CDA-93205E99EC1C,00000000)/VenHw(964E5B22-
6459-11D2-8E39-00A0C969723B,00000000)
  Optional- Y
Option: 04. Variable: Boot0004
  Desc    - UEFI BootManagerMenuApp
  DevPath - Fv(6D99E806-3D38-42C2-A095-5F4300BFD7DC)/FvFile(EEC25BDC-67F2-4D95-B
1D5-F81B2039D11D)/BootManagerMenuApp
  Optional- Y
Option: 05. Variable: Boot0000
  Desc    - UEFI Enter Setup
  DevPath - Fv(6D99E806-3D38-42C2-A095-5F4300BFD7DC)/FvFile(462CAA21-7614-4503-8
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 NV+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 NV+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 shell*.efi
Directory of: FS0:\

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

```
FS0:\> Dir
```

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

Result

```
  Optional- Y
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:\> _
```

Exit QEMU

Lesson Objective

- Run UEFI Shell in QEMU
- Run UEFI Shell Commands
- Run UEFI Shell Scripts

tianocore

Questions?

Return to Main Training Page

Return to Training Table of contents for next presentation link

tianocore

tianocore

www.tianocore.org

ACKNOWLEDGEMENTS