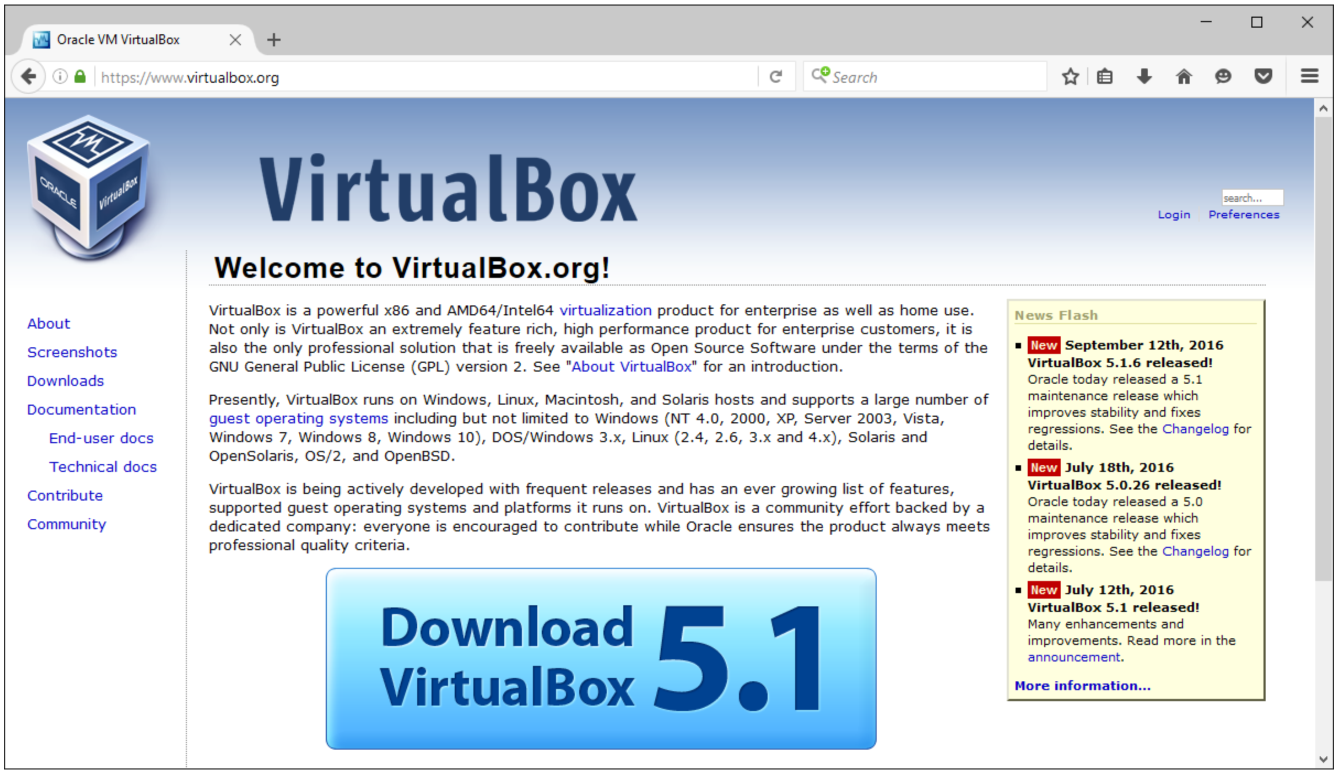
Installing Oracle® VirtualBox on Windows 10 and Creating a Starter Boot Disk for a Custom Operating System

David J. Walling
Twitter: @davidjwalling
September 14, 2016

This document describes steps for downloading and installing Oracle VirtualBox for Windows. VirtualBox is a virtual machine (VM) manager supported on a wide variety of host operating systems and able to run many guest operating systems. As of this writing, VirtualBox 5.1.6 is the current version. At https://www.virtualbox.org, click on the large Download VirtualBox 5.1 button shown below.



Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "About VirtualBox" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of guest operating systems including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

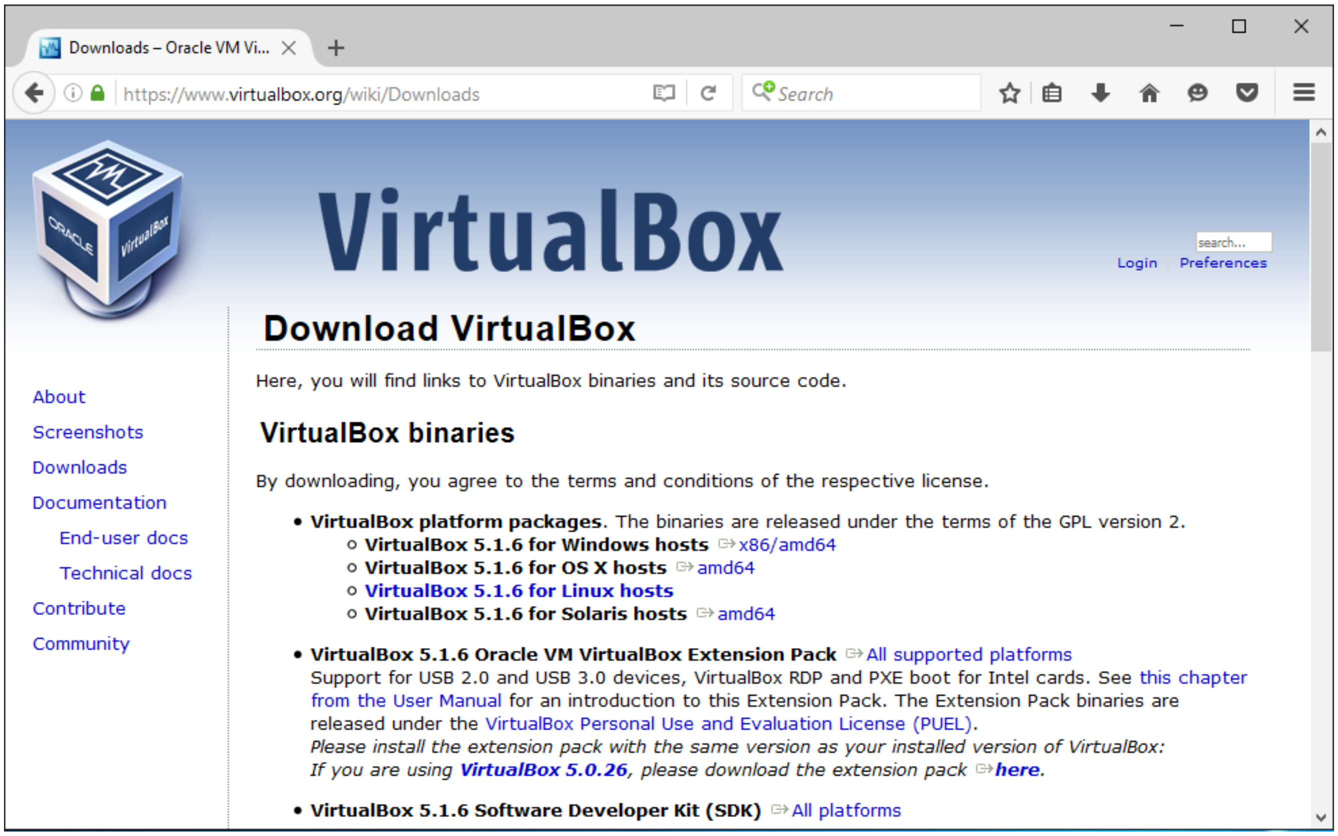
VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

News Flash

- New **September 12th, 2016**
  VirtualBox 5.1.6 released!
  Oracle today released a 5.1 maintenance release which improves stability and fixes regressions. See the Changelog for details.
- New **July 18th, 2016**
  VirtualBox 5.0.26 released!
  Oracle today released a 5.0 maintenance release which improves stability and fixes regressions. See the Changelog for details.
- New **July 12th, 2016**
  VirtualBox 5.1 released!
  Many enhancements and improvements. Read more in the announcement.

More information...

VirtualBox supports several host operating systems. In our case, we are planning to run VirtualBox on Windows 10, running on a 64-bit Intel or AMD processor. Select the “x86/amd64” link for “VirtualBox 5.1.6 for Windows hosts.”


Downloads – Oracle VM Vi...

https://www.virtualbox.org/wiki/Downloads

VirtualBox

Login Preferences

About
Screenshots
Downloads
Documentation
End-user docs
Technical docs
Contribute
Community

Download VirtualBox

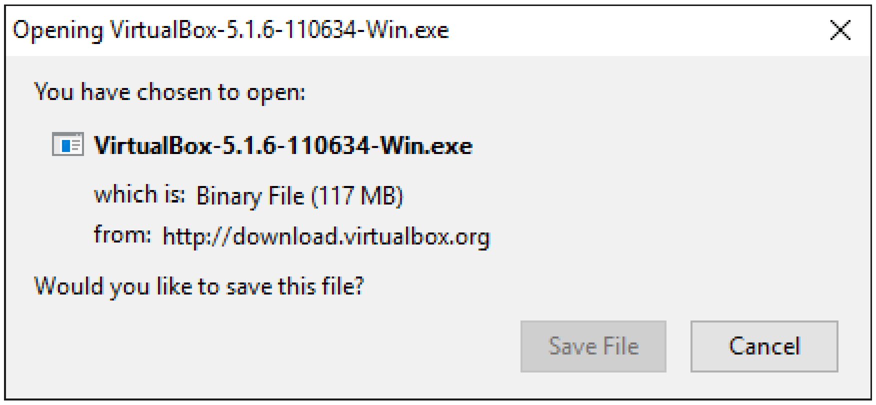
Here, you will find links to VirtualBox binaries and its source code.

VirtualBox binaries

By downloading, you agree to the terms and conditions of the respective license.

- **VirtualBox platform packages**. The binaries are released under the terms of the GPL version 2.
  - **VirtualBox 5.1.6 for Windows hosts** x86/amd64
  - **VirtualBox 5.1.6 for OS X hosts** amd64
  - **VirtualBox 5.1.6 for Linux hosts**
  - **VirtualBox 5.1.6 for Solaris hosts** amd64
- **VirtualBox 5.1.6 Oracle VM VirtualBox Extension Pack** All supported platforms
  Support for USB 2.0 and USB 3.0 devices, VirtualBox RDP and PXE boot for Intel cards. See this chapter from the User Manual for an introduction to this Extension Pack. The Extension Pack binaries are released under the VirtualBox Personal Use and Evaluation License (PUEL).
  Please install the extension pack with the same version as your installed version of VirtualBox:
  If you are using ***VirtualBox 5.0.26****, please download the extension pack* ***here****.*
- **VirtualBox 5.1.6 Software Developer Kit (SDK)** All platforms

If prompted to choose whether to save or run the file, select “Save File”.


Opening VirtualBox-5.1.6-110634-Win.exe

You have chosen to open:

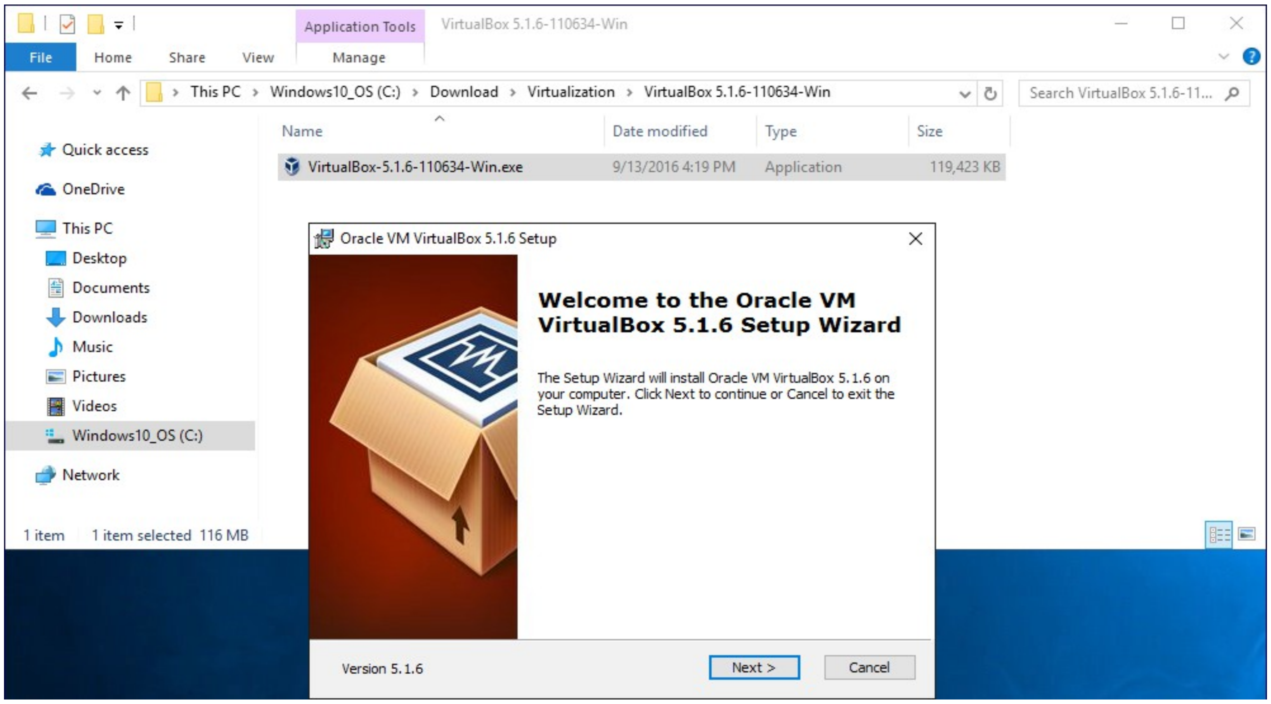
VirtualBox-5.1.6-110634-Win.exe

which is: Binary File (117 MB)

from: http://download.virtualbox.org

Would you like to save this file?

After downloading, click on the downloaded executable, VirtualBox-5.1.6-110634-Win.exe to start the installation process. At the welcome screen, click “Next” to start the installation process.


Oracle VM VirtualBox 5.1.6 Setup

Welcome to the Oracle VM VirtualBox 5.1.6 Setup Wizard

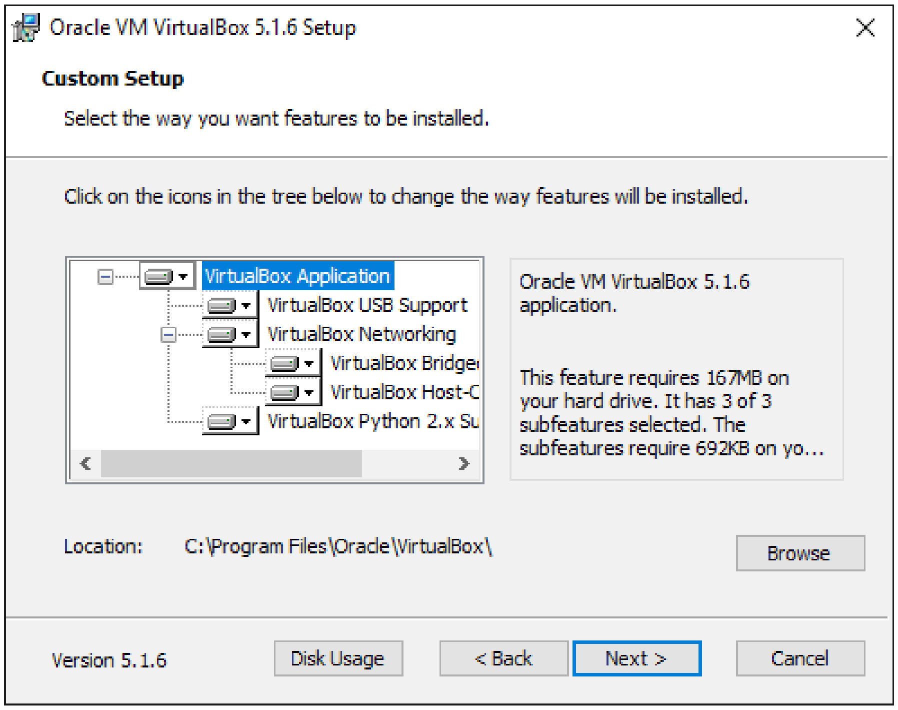
The Setup Wizard will install Oracle VM VirtualBox 5.1.6 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Version 5.1.6

Name	Date modified	Type	Size
VirtualBox-5.1.6-110634-Win.exe	9/13/2016 4:19 PM	Application	119,423 KB

This PC > Windows10_OS (C:) > Download > Virtualization > VirtualBox 5.1.6-110634-Win

Confirm the options to install. In this case, we will accept the default settings to install the complete package. Verify the location for your installation. Use the "Browse" button if you want to change the location.


Oracle VM VirtualBox 5.1.6 Setup

Custom Setup

Select the way you want features to be installed.

Click on the icons in the tree below to change the way features will be installed.

- VirtualBox Application
  - VirtualBox USB Support
  - VirtualBox Networking
    - VirtualBox Bridged
    - VirtualBox Host-C
  - VirtualBox Python 2.x Su

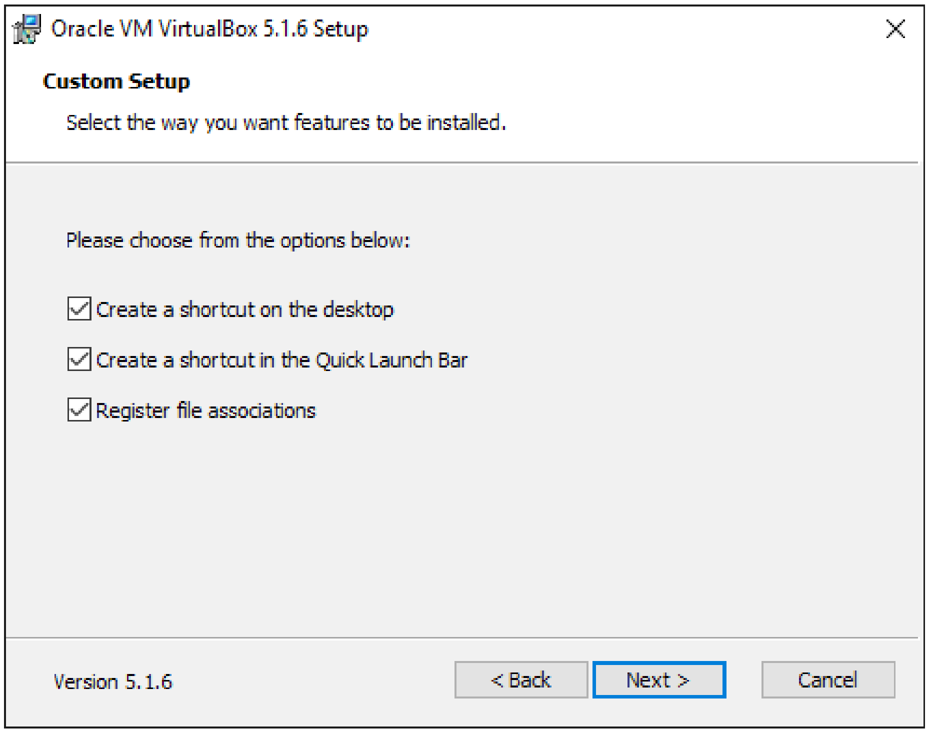
Oracle VM VirtualBox 5.1.6 application.

This feature requires 167MB on your hard drive. It has 3 of 3 subfeatures selected. The subfeatures require 692KB on yo...

Location: C:\Program Files\Oracle\VirtualBox\

Version 5.1.6

Click "Next" to advance to the next setup dialog. Customize the options on this screen as you like. We will accept the default settings.


Oracle VM VirtualBox 5.1.6 Setup

Custom Setup

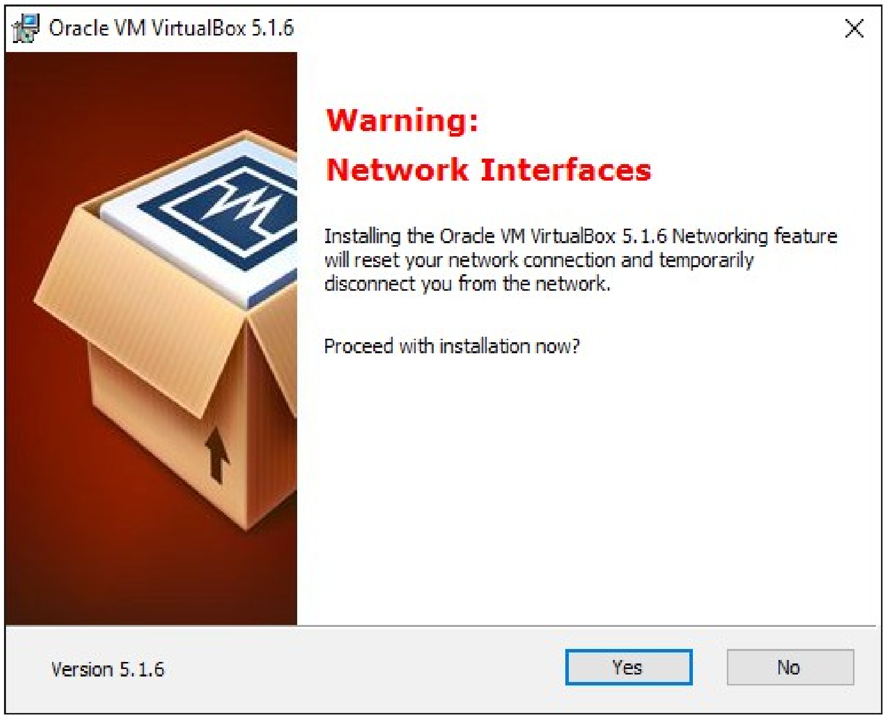
Select the way you want features to be installed.

Please choose from the options below:

- [✓] Create a shortcut on the desktop
- [✓] Create a shortcut in the Quick Launch Bar
- [✓] Register file associations

Version 5.1.6

Click “Next” to advance to next setup dialog. Note the warning shown on this dialog about resetting the network connection. Complete any pending network activity before continuing.


Oracle VM VirtualBox 5.1.6

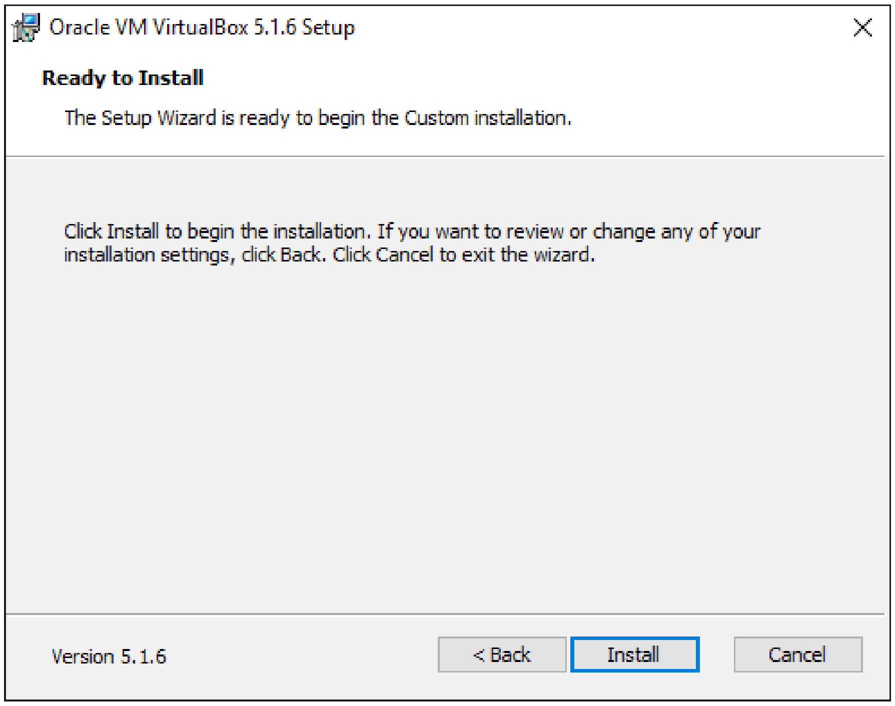
Warning:
Network Interfaces

Installing the Oracle VM VirtualBox 5.1.6 Networking feature will reset your network connection and temporarily disconnect you from the network.

Proceed with installation now?

Version 5.1.6

Click “Yes” to continue with the installation. The “Ready to Install” dialog allows you to return to any prior dialog using the “Back” button before finalizing your selections and completing the installation.



Oracle VM VirtualBox 5.1.6 Setup

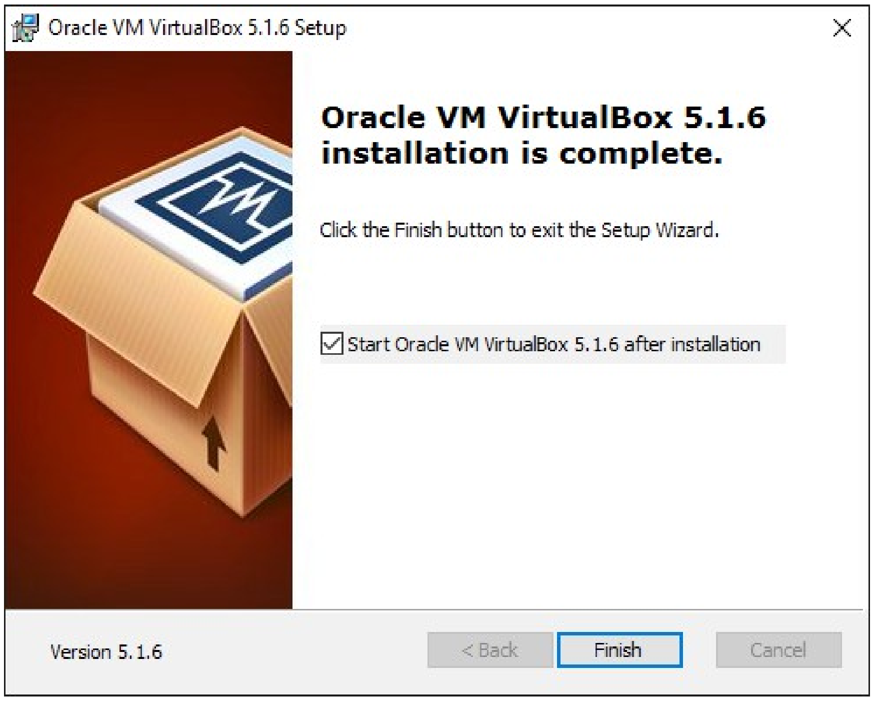
Ready to Install

The Setup Wizard is ready to begin the Custom installation.

Click Install to begin the installation. If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

Version 5.1.6

Click "Install" when you are ready to finish the installation. Once the installation completes, indicate by the check-box setting whether you want to start Oracle VM VIrtualBox after the installation.



Oracle VM VirtualBox 5.1.6 Setup

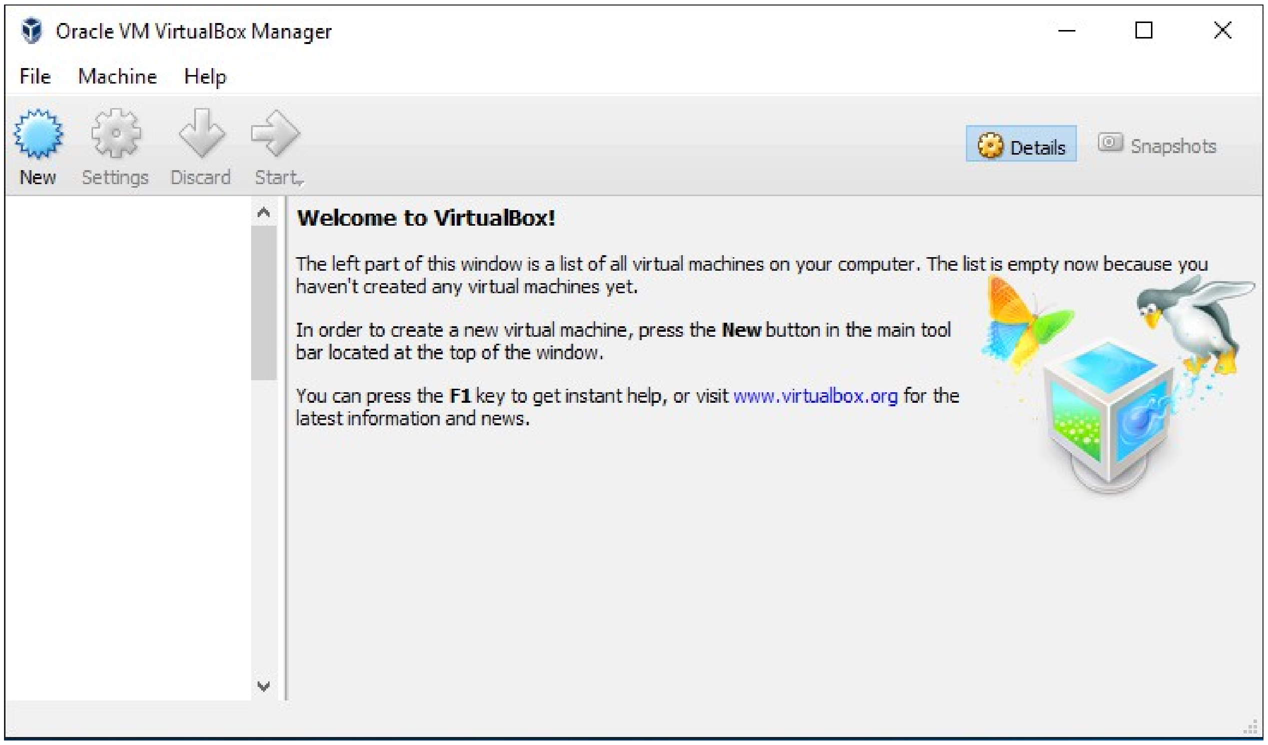
Oracle VM VirtualBox 5.1.6 installation is complete.

Click the Finish button to exit the Setup Wizard.

[✓] Start Oracle VM VirtualBox 5.1.6 after installation

Version 5.1.6

Click "Finish" to complete the installation process. When started, VirtualBox displays the manager window which will list our VM instance in the left pane and details in the right pane.



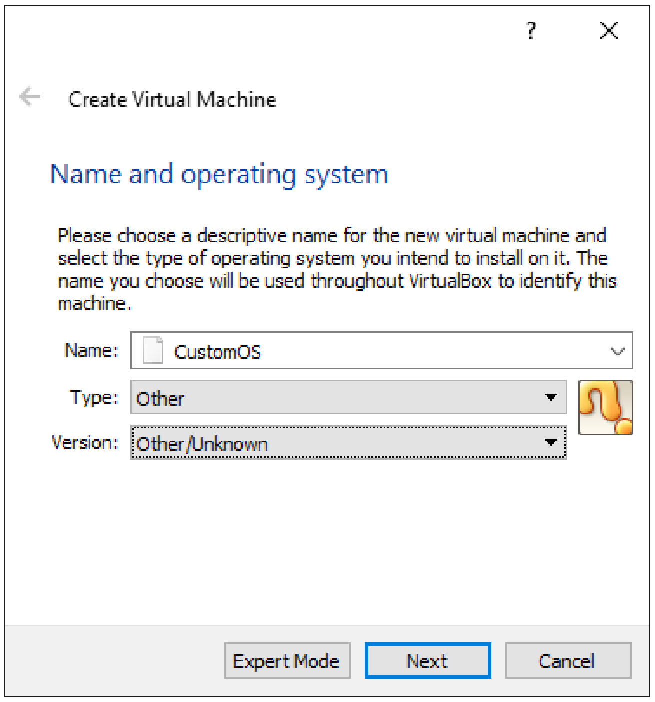
Welcome to VirtualBox!

The left part of this window is a list of all virtual machines on your computer. The list is empty now because you haven't created any virtual machines yet.

In order to create a new virtual machine, press the **New** button in the main tool bar located at the top of the window.

You can press the **F1** key to get instant help, or visit www.virtualbox.org for the latest information and news.

Now we can define a VM for our custom operating system. Click "New" to open the "Name and operating system" dialog. We'll enter a name for the VM, "CustomOS", select "Other" as the Type and "Other/Unknown" as the Version.



Name and operating system

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

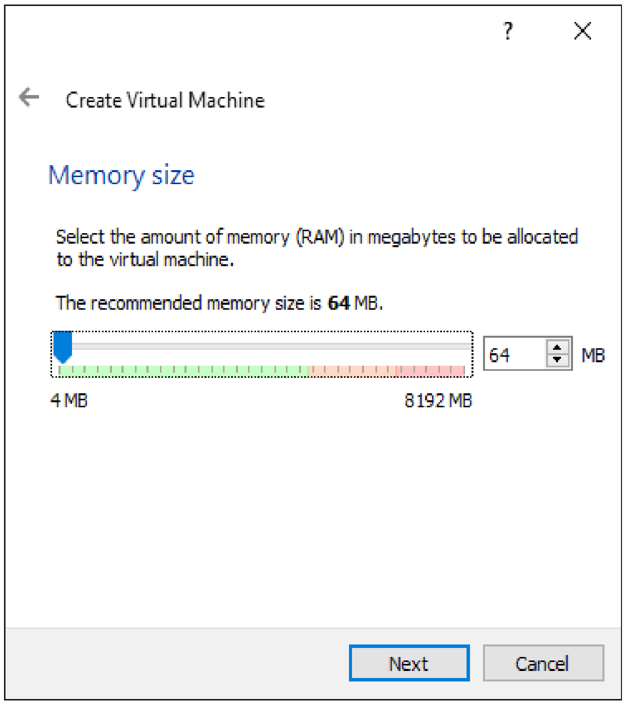
Name: CustomOS

Type: Other

Version: Other/Unknown

Click, "Next" to proceed to the next dialog.

In the “Memory size” dialog, we accept the default size of 64 MB. It will be a little while before we need more storage than that.

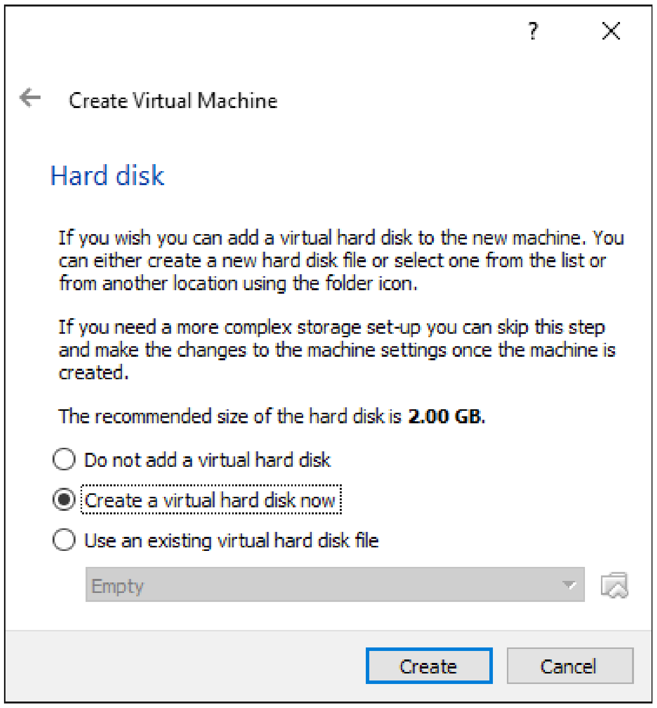

Create Virtual Machine

Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **64** MB.

Click “Next” to continue to the next dialog. In the “Hard disk” dialog, select the “Create a virtual hard disk now” radio-button option. We'll create a hard disk, but it won't be used right away.


Create Virtual Machine

Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

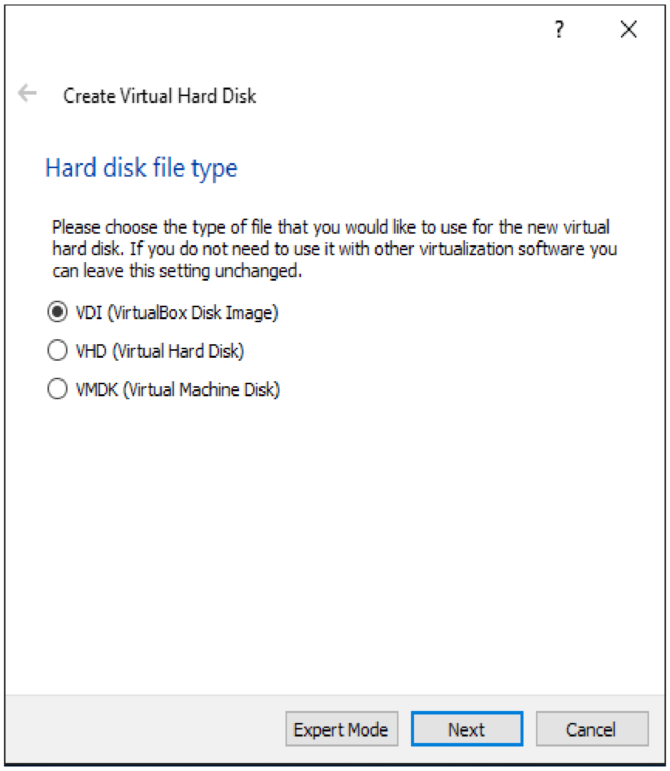
If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **2.00 GB**.

Do not add a virtual hard disk [✓ Create a virtual hard disk now] Use an existing virtual hard disk file

Click “Create” to continue.

In the “Hard disk file type” dialog, select the “VDI (VirtualBox Disk Image)” type.

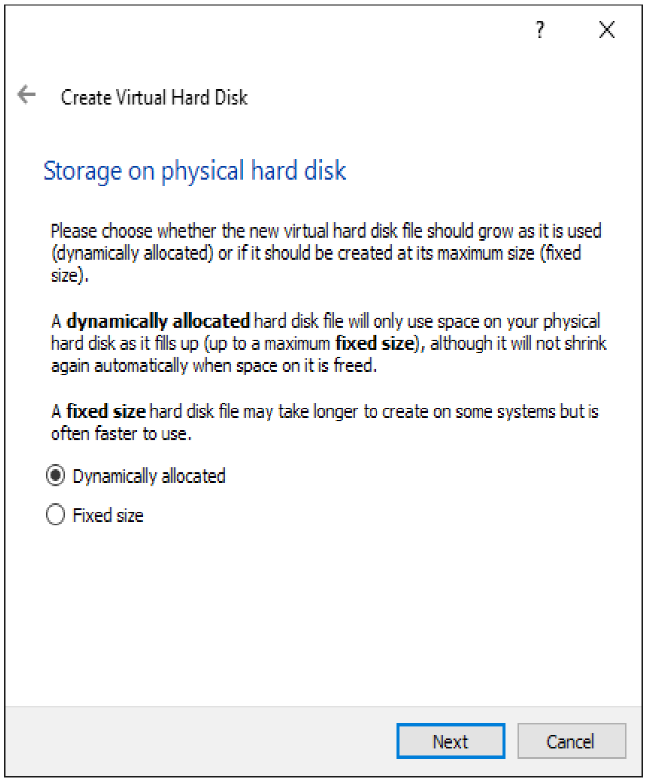


Hard disk file type

Please choose the type of file that you would like to use for the new virtual hard disk. If you do not need to use it with other virtualization software you can leave this setting unchanged.

- [✓ VDI (VirtualBox Disk Image)]
- VHD (Virtual Hard Disk)
- VMDK (Virtual Machine Disk)

Click “Next”. In the “Storage on physical hard disk” dialog, select “Dynamically allocated.”



Storage on physical hard disk

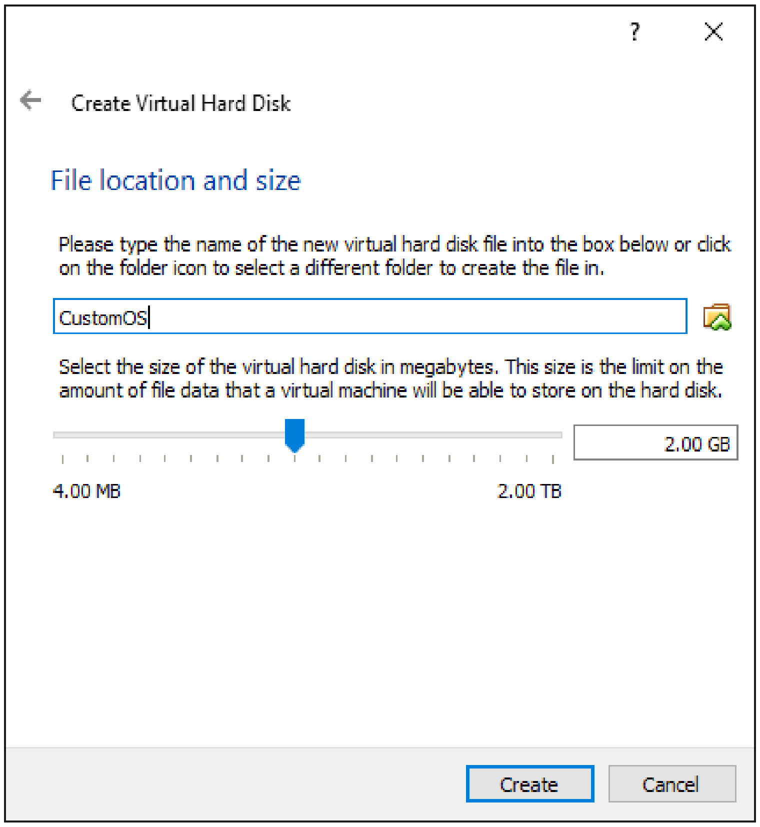
Please choose whether the new virtual hard disk file should grow as it is used (dynamically allocated) or if it should be created at its maximum size (fixed size).

A **dynamically allocated** hard disk file will only use space on your physical hard disk as it fills up (up to a maximum **fixed size**), although it will not shrink again automatically when space on it is freed.

A **fixed size** hard disk file may take longer to create on some systems but is often faster to use.

- [✓ Dynamically allocated]
- Fixed size

Click “Next”. In the “File location and size” dialog, we provide a name for the file (the extension is omitted), and take the default recommended disk size of 2 GB.



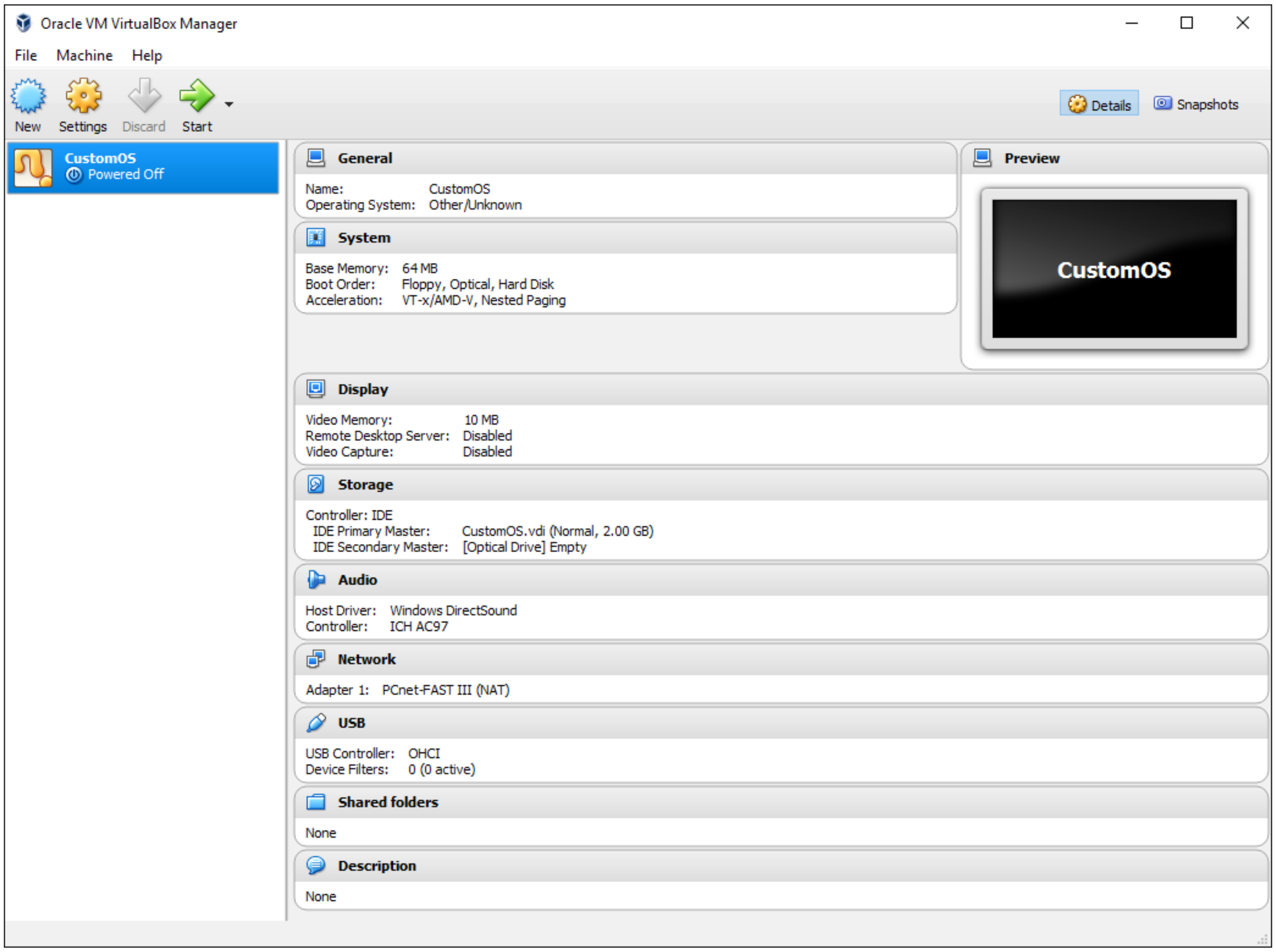
Please type the name of the new virtual hard disk file into the box below or click on the folder icon to select a different folder to create the file in.

CustomOS

Select the size of the virtual hard disk in megabytes. This size is the limit on the amount of file data that a virtual machine will be able to store on the hard disk.

2.00 GB

Clicking “Create” here will create the hard disk file for the VM and return you to the VM Manager screen. The new “CustomOS” VM appears now in the left pane. Details about our VM are shown in the right pane.



General
Name: CustomOS
Operating System: Other/Unknown

System
Base Memory: 64 MB
Boot Order: Floppy, Optical, Hard Disk
Acceleration: VT-x/AMD-V, Nested Paging

Display
Video Memory: 10 MB
Remote Desktop Server: Disabled
Video Capture: Disabled

Storage
Controller: IDE
IDE Primary Master: CustomOS.vdi (Normal, 2.00 GB)
IDE Secondary Master: [Optical Drive] Empty

Audio
Host Driver: Windows DirectSound
Controller: ICH AC97

Network
Adapter 1: PCnet-FAST III (NAT)

USB
USB Controller: OHCI
Device Filters: 0 (0 active)

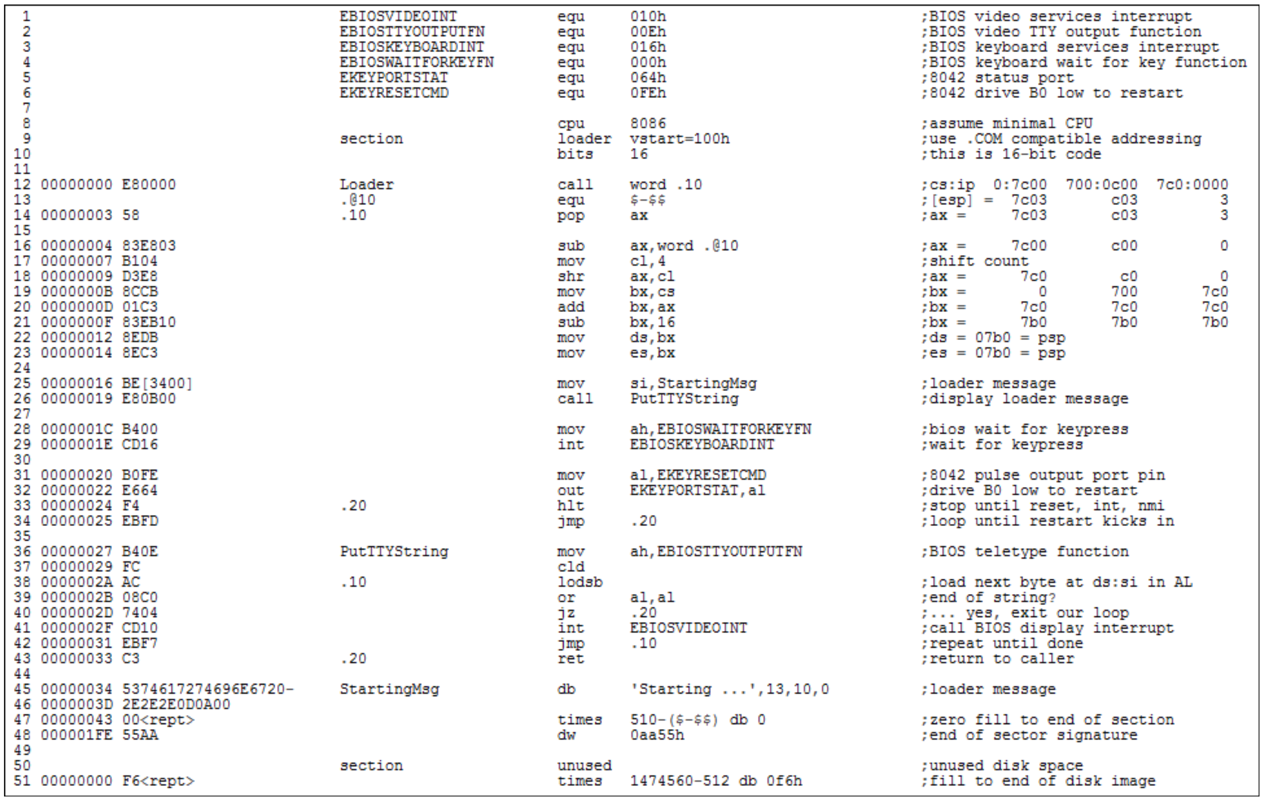
Shared folders
None

Description
None

One thing to notice in the details shown in the right pane is the Boot Order within the "System" section. The boot order is listed as "Floppy, Optical, Hard Disk". This is configurable. But, we are going to leave this boot order unchanged and take advantage of the fact that booting from a "floppy" disk is first in our boot order. Floppy disk images are quite small, relative to hard disk sizes, that is. And for a custom operating system (read "simple"), this disk image type will be a convenient size to work with.

So, we want to define a floppy disk image with our custom operating system on it. Let's change direction for a few minutes, then, and write a quick assembly-language program to generate a working boot disk image file.

I'm using Netwide Assembler (NASM) to assemble the code examples that follow. The last time I checked, this assembler was available at www.nasm.us. I'm not endorsing it, I just like it. Here's the assembly listing.



```
 1                                EBIOSVIDEOINT       equ     010h                    ;BIOS video services interrupt
 2                                EBIOSTTYOUTPUTFN    equ     00Eh                    ;BIOS video TTY output function
 3                                EBIOSKEYBOARDINT    equ     016h                    ;BIOS keyboard services interrupt
 4                                EBIOSWAITFORKEYFN   equ     000h                    ;BIOS keyboard wait for key function
 5                                EKEYPORTSTAT        equ     064h                    ;8042 status port
 6                                EKEYRESETCMD        equ     0FEh                    ;8042 drive B0 low to restart
 7
 8                                                    cpu     8086                    ;assume minimal CPU
 9                                section             loader  vstart=100h             ;use .COM compatible addressing
10                                                    bits    16                      ;this is 16-bit code
11
12 00000000 E80000                Loader              call    word .10                ;cs:ip  0:7c00  700:0c00  7c0:0000
13                                .@10                equ     $-$$                    ;[esp] =  7c03       c03         3
14 00000003 58                    .10                 pop     ax                      ;ax =     7c03       c03         3
15
16 00000004 83E803                                    sub     ax,word .@10            ;ax =     7c00       c00         0
17 00000007 B104                                      mov     cl,4                    ;shift count
18 00000009 D3E8                                      shr     ax,cl                   ;ax =      7c0        c0         0
19 0000000B 8CCB                                      mov     bx,cs                   ;bx =        0       700       7c0
20 0000000D 01C3                                      add     bx,ax                   ;bx =      7c0       7c0       7c0
21 0000000F 83EB10                                    sub     bx,16                   ;bx =      7b0       7b0       7b0
22 00000012 8EDB                                      mov     ds,bx                   ;ds = 07b0 = psp
23 00000014 8EC3                                      mov     es,bx                   ;es = 07b0 = psp
24
25 00000016 BE[3400]                                  mov     si,StartingMsg          ;loader message
26 00000019 E80B00                                    call    PutTTYString            ;display loader message
27
28 0000001C B400                                      mov     ah,EBIOSWAITFORKEYFN    ;bios wait for keypress
29 0000001E CD16                                      int     EBIOSKEYBOARDINT        ;wait for keypress
30
31 00000020 B0FE                                      mov     al,EKEYRESETCMD         ;8042 pulse output port pin
32 00000022 E664                                      out     EKEYPORTSTAT,al         ;drive B0 low to restart
33 00000024 F4                    .20                 hlt                             ;stop until reset, int, nmi
34 00000025 EBFD                                      jmp     .20                     ;loop until restart kicks in
35
36 00000027 B40E                  PutTTYString        mov     ah,EBIOSTTYOUTPUTFN     ;BIOS teletype function
37 00000029 FC                                        cld
38 0000002A AC                    .10                 lodsb                           ;load next byte at ds:si in AL
39 0000002B 08C0                                      or      al,al                   ;end of string?
40 0000002D 7404                                      jz      .20                     ;... yes, exit our loop
41 0000002F CD10                                      int     EBIOSVIDEOINT           ;call BIOS display interrupt
42 00000031 EBF7                                      jmp     .10                     ;repeat until done
43 00000033 C3                    .20                 ret                             ;return to caller
44
45 00000034 5374617274696E6720-   StartingMsg         db      'Starting ...',13,10,0  ;loader message
46 0000003D 2E2E2E0D0A00
47 00000043 00<rept>                                  times   510-($-$$) db 0         ;zero fill to end of section
48 000001FE 55AA                                      dw      0aa55h                  ;end of sector signature
49
50                                section             unused                          ;unused disk space
51 00000000 F6<rept>                                  times   1474560-512 db 0f6h     ;fill to end of disk image
```

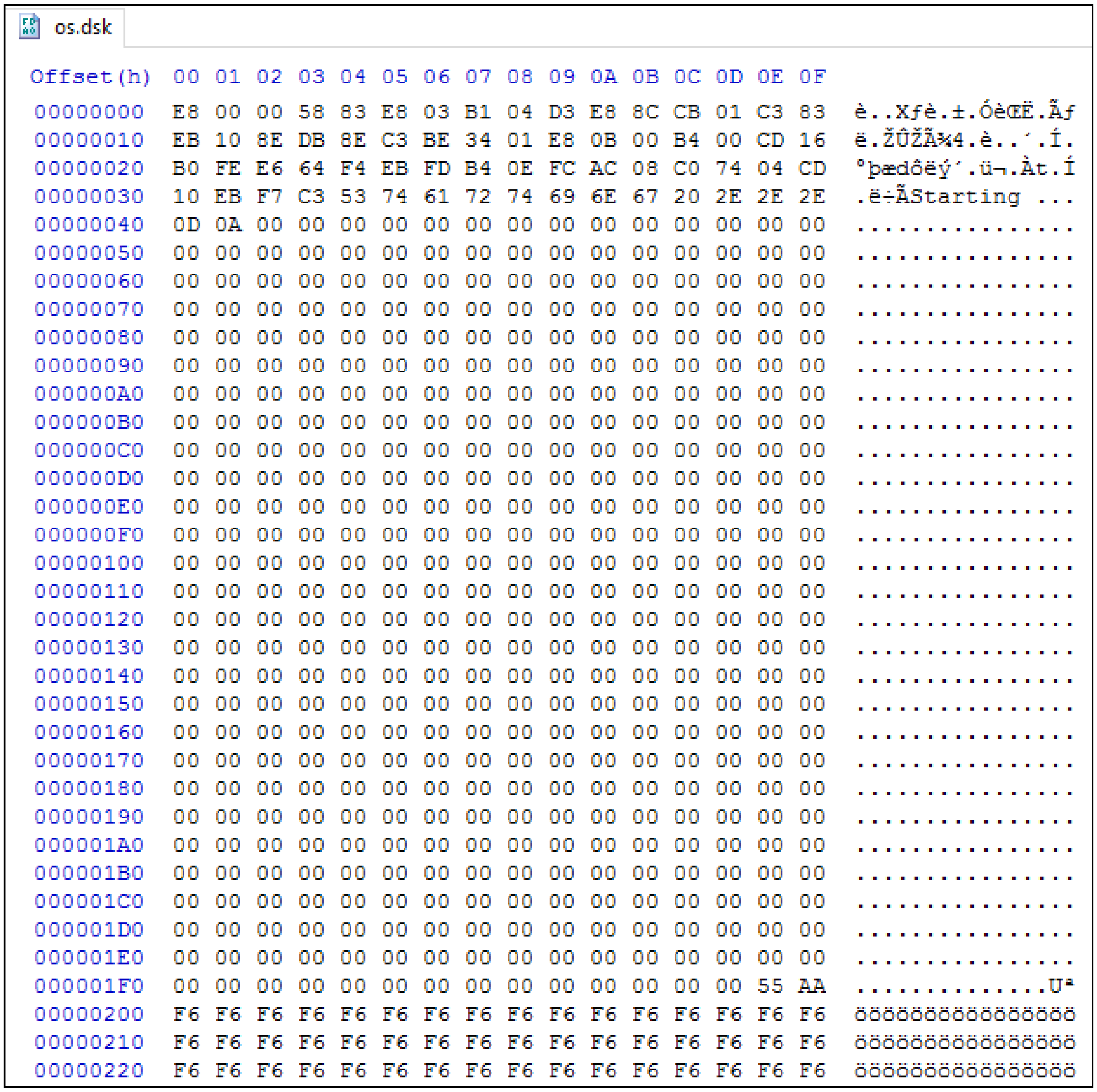
Defining symbolic equates at the top we helps us remember what the values are for and that we use consistent values. The “cpu” directive on line 8 informs the assembler that we do not want any instructions incompatible with an 8086 CPU generated at this stage. The “bits” directive informs the assembler that our target is a system that expects to be running in 16-bit mode at start-up. The “loader” section is defined with a virtual starting address of 100h. This is so that this code, once it becomes an actual “file” on our disk image, will behave like a DOS “.COM” file when loaded by the BIOS.

When our VM starts, it will copy the first sector (512 bytes) of our floppy disk image to the VM's virtual memory address 0x7C00 and start execution at offset zero (line 12 in our listing). At this point, we really have no idea what values our segment registers will hold. So, we make no assumptions other than that our instruction pointer (IP) contains a valid address. If it didn't we wouldn't be here. The instructions on lines 12 through 23 establish proper data segment address-ability to our program data by making sure that DS and ES both point to the Program Segment Prefix (PSP), which is 100h bytes before the start of the program code. How this works is described by the inline comments.

With DS setup, we can properly address “StartingMsg” using DS:SI. The PutTTYString subroutine does exactly that, using the BIOS video interrupt. After displaying the message “Starting ...” to the console, we call another BIOS function to wait for a key-press. This gives us time to admire the message we displayed. Now we can force a reboot by driving the B0 line of our 8042 keyboard controller low and wait for the hardware reset to kick in.

After the “Starting ...” message, we have some null filler to the end of the 512 byte sector, which ends with the expected 0x55 0xAA marker. Then the remainder of the 1.44 MB disk image is initialized to 0xF6, which indicates never-used storage.

The source file was stored as “os.asm”. The code is assembled using the command “nasm os.asm -f bin -o os.dsk -l os.lst”. The os.lst listing file is what we looked at above. The “-f” assembly parameter tells nasm to output a flat-form binary file. We can look at the os.dsk output file in a hex editor. I use HxD Hex Editor.



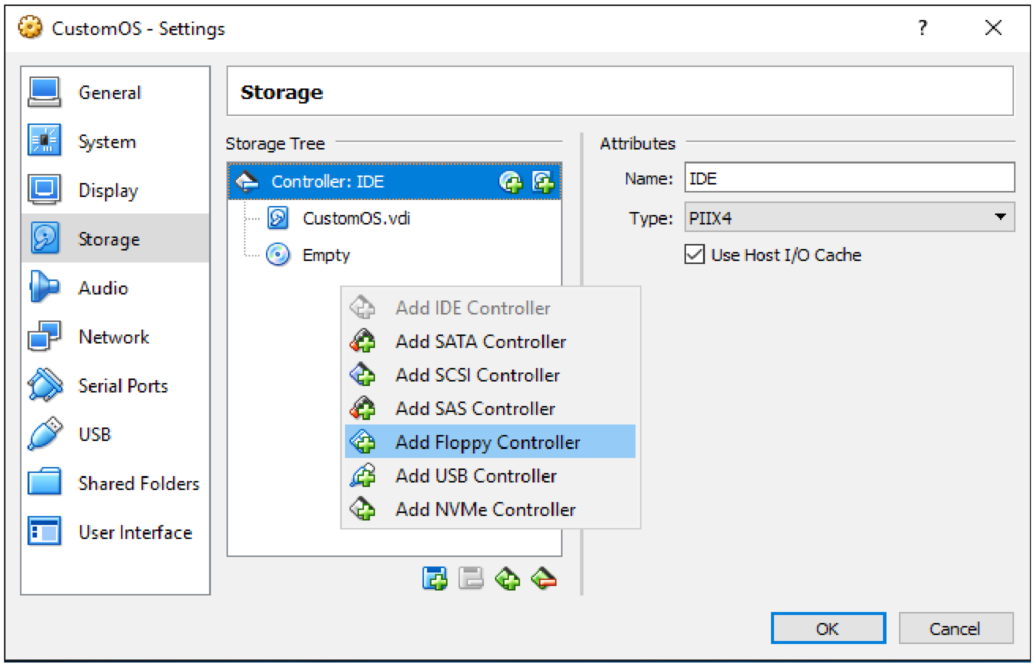
os.dsk

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  E8 00 00 58 83 E8 03 B1 04 D3 E8 8C CB 01 C3 83  è..Xƒè.±.ÓèŒË.Ãƒ
00000010  EB 10 8E DB 8E C3 BE 34 01 E8 0B 00 B4 00 CD 16  ë.ŽÛŽÃ¾4.è..´.Í.
00000020  B0 FE E6 64 F4 EB FD B4 0E FC AC 08 C0 74 04 CD  °þædôëý´.ü¬.Àt.Í
00000030  10 EB F7 C3 53 74 61 72 74 69 6E 67 20 2E 2E 2E  .ë÷ÃStarting ...
00000040  0D 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000060  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000070  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000100  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000110  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000120  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000130  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000140  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000150  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000170  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000180  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001D0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001E0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000001F0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA  ..............Uª
00000200  F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6  öööööööööööööööö
00000210  F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6  öööööööööööööööö
00000220  F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6 F6  öööööööööööööööö
```

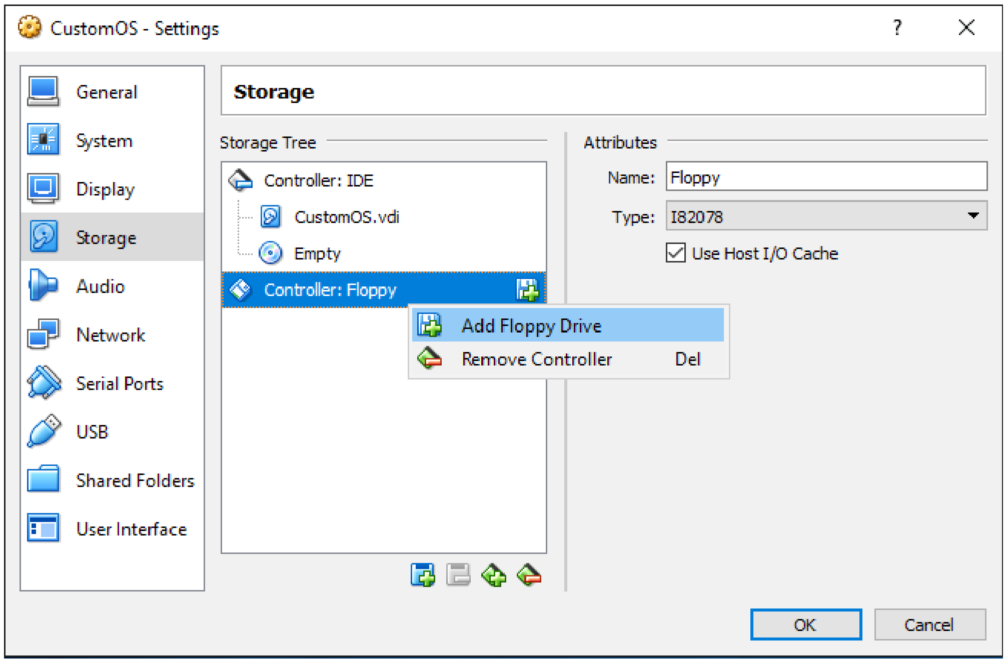
Note the end-of-sector marker at offset 0x1FE and the start of unused disk filler at 0x200. This is not at all what a dump of a real boot disk would look like. We're completely bypassing a proper boot sector, file allocation table and disk directory. But that's fine for now. VirtualBox doesn't care.

Now we'll update our VM settings to define a diskette controller, a floppy disk drive and point to the floppy disk image file we just created.



Name: IDE
Type: PIIX4
☑ Use Host I/O Cache

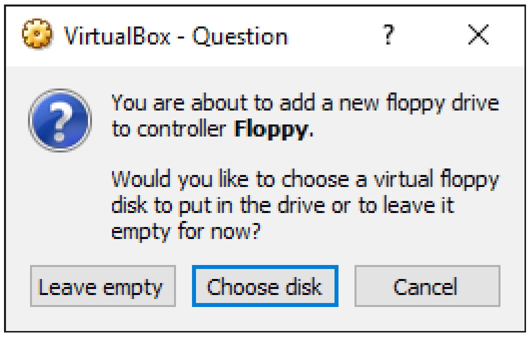
Back in the VM Manager, click on the "Settings" toolbar icon to open the "Settings" dialog for our CustomOS VM. Click on the "Storage" option to display storage settings. Right click in the "Storage Tree" window and select the "Add Floppy Controller" menu option. This will add a line in the Storage Tree labeled "Controller: Floppy".



Name: Floppy
Type: I82078
☑ Use Host I/O Cache

Right mouse click on the new Controller entry and select "Add Floppy Drive".

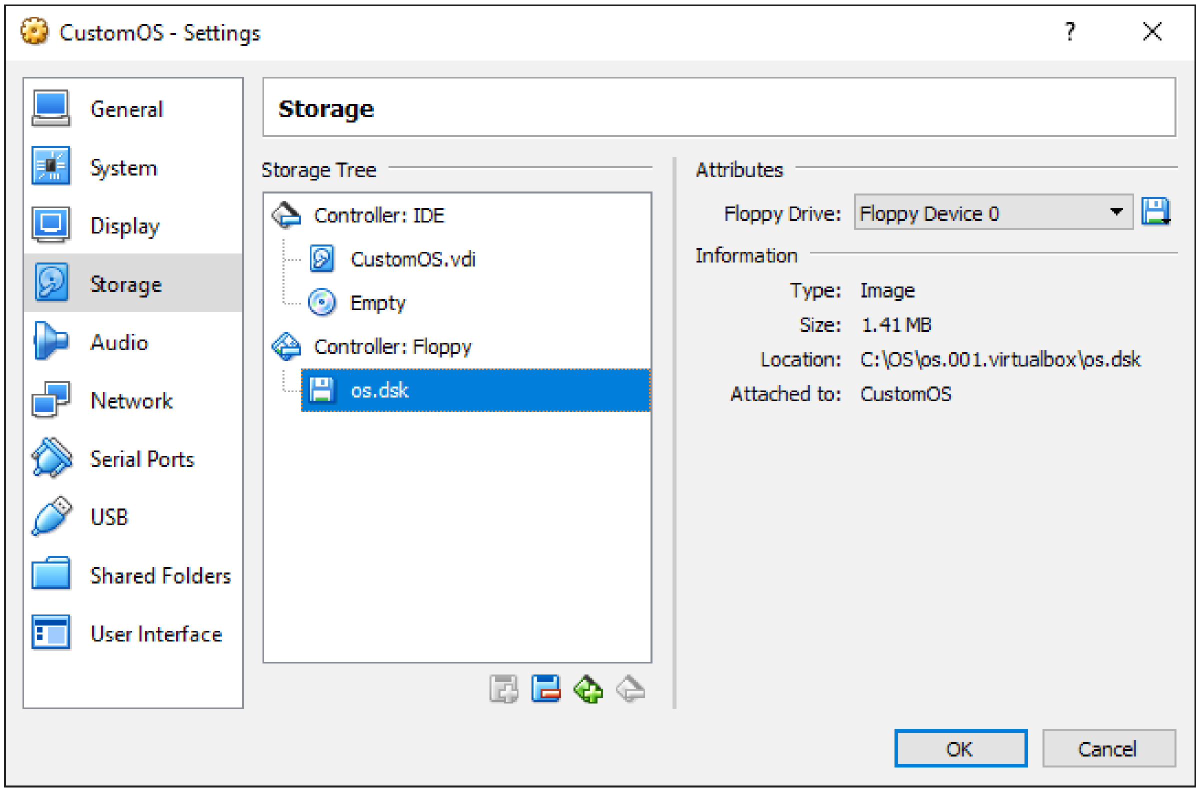
The “Question” dialog - I love that name - opens and asks if you want to choose an existing floppy disk file or leave that empty. We select “Choose disk”.



You are about to add a new floppy drive to controller **Floppy**.

Would you like to choose a virtual floppy disk to put in the drive or to leave it empty for now?

Navigate to the location of the os.dsk file created above and select it. The Storage Tree is now updated to show our selected file.

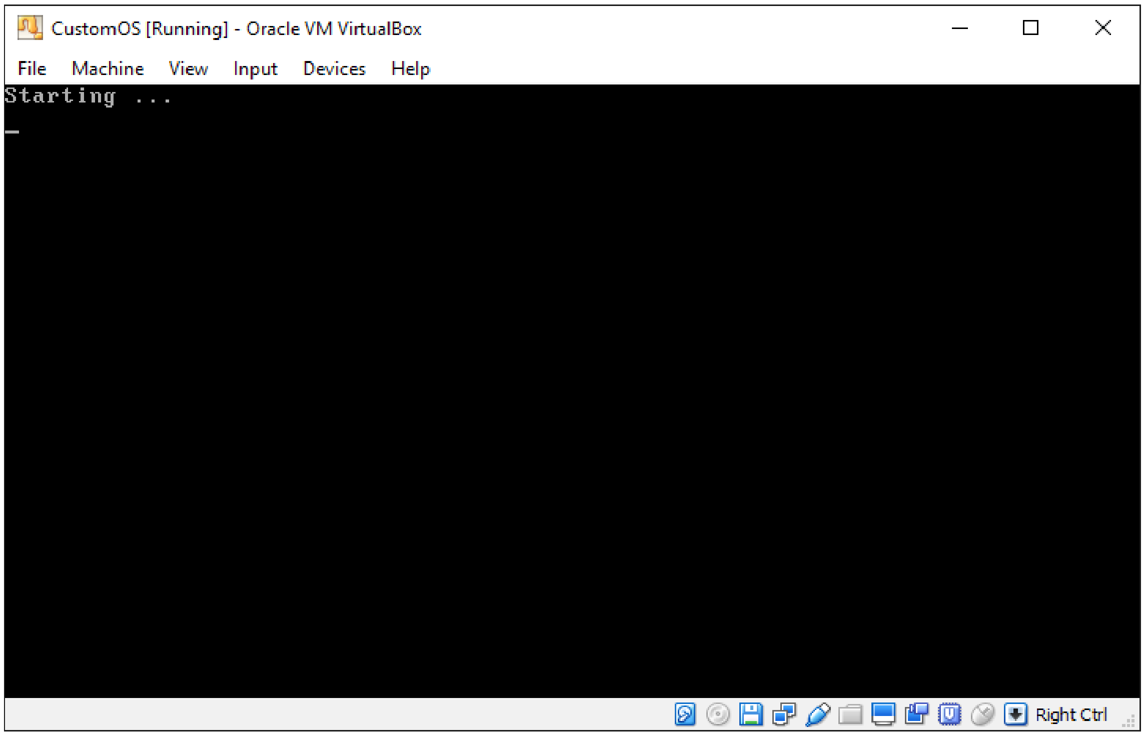


Floppy Drive: Floppy Device 0

Type: Image
Size: 1.41 MB
Location: C:\OS\os.001.virtualbox\os.dsk
Attached to: CustomOS

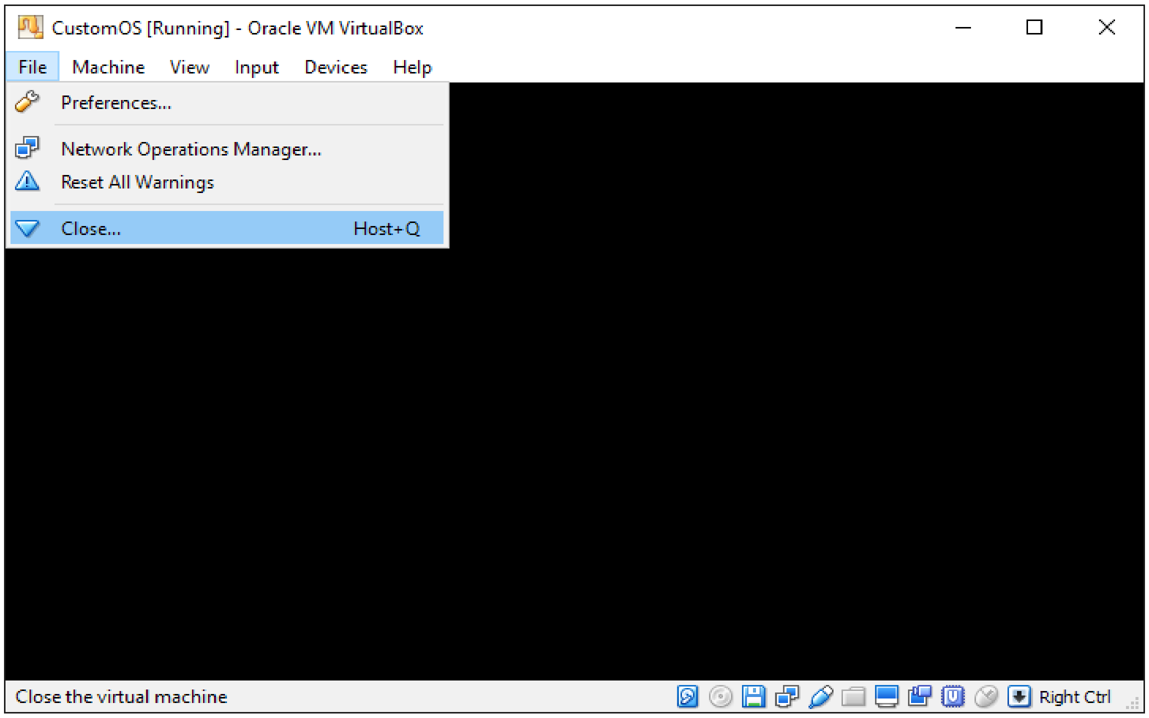
Now, yes, I did notice that the size of our floppy drive is given here as “1.41MB”. That might surprise someone old enough to remember when these disks were called “1.44MB” floppy disks. But the “1.44” was always a misnomer. The true unformatted size of a “1.44MB” floppy was 1,474,560 bytes, which is 1,440 KB or 1.40625 MB, precisely. VirtualBox is correct again!

Click “OK” to save the updated VM settings and restart the VM.

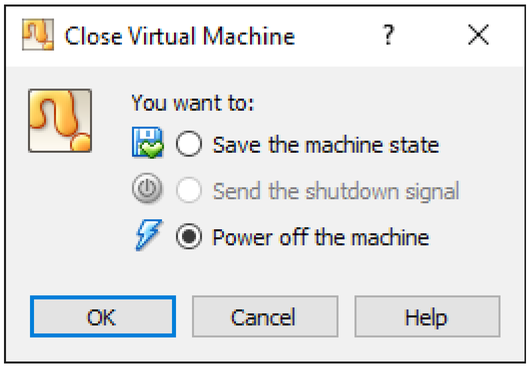

CustomOS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Right Ctrl

```
Starting ...
_
```

If we click our mouse into the VM window and press a key, the operating system will restart itself as we programmed it to do. VirtualBox has reserved the "Right Ctrl" key combination for us to recover our mouse and regain control of it in our host operating system environment. Now all that is left to do is shut down. Use the "File"|"Close" menu options to do that.


CustomOS [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Preferences...
Network Operations Manager...
Reset All Warnings
Close... Host+Q
Close the virtual machine
Right Ctrl

When the “File”|”Close” option is selected, VirtualBox prompts us to decide whether to save the machine state or simply power off the machine. Note that VirtualBox recognizes that our custom operating system doesn't accept a shutdown signal that originates from the VM manager.


Close Virtual Machine

You want to:

○ Save the machine state

○ Send the shutdown signal

◉ Power off the machine

Select “Power off the machine” and click “OK”. This will stop our VM instance and return us to the VM manager window.

Well, now, as a learning experience, we can build out functionality in our custom operating system. We'll take that up in subsequent documents.