Bahir Dar University

Faculty of Computing

Department of Software Engineering

OSSP Individual Assignment

AN OVERVIEW OF OPERATING SYSTEM - OpenVMS: HISTORY, FEATURES, REQUIREMENTS, AND FUTURE TRENDS

Full Name – Hiywot Teshome
Student Id – BDU1601788
Section – A
System Call – recvfrom()
Operating system - OpenVMS

Submitted To: Lecturer Wondimu B.

Table of Contents

AN OVERVIEW OF OpenVMS: HISTORY, FEATURES, REQUIREMENTS, AND FUTURE TRENDS

INTRODUCTION TO OpenVMS

Background

OpenVMS, or Open Virtual Memory System, often referred to as just VMS, is a multi-user, multiprocessing and virtual memory-based operating system. It is designed to support time-sharing, batch processing, transaction processing and workstation applications. Customers using OpenVMS include banks and financial services, hospitals and healthcare, telecommunications operators, network information services, and industrial manufacturers. During the 1990s and 2000s, there were approximately half a million VMS systems in operation worldwide.

It was first announced by Digital Equipment Corporation (DEC) as VAX/VMS in 1977. Therefore, it is sometimes referred to as DEC VAX VMS or DEC VMS operating system. It was specifically designed to run on computers with the VAX (Virtual Address eXtension) architecture, a 32-bit computing system that succeeded 16-bit PDP-11 systems.

It was first announced by Digital Equipment Corporation (DEC) as **VAX/VMS in 1977. which was later acquired by Compaq and HP.** HP later discontinued its support for OpenVMS for some hardware, so in response to that, VMS Software, Inc. (VSI) was formed and acquired the rights to develop, distribute, and support it.

Evolution (History)

Even though OpenVMS OS is one of the oldest operating system created but, it is from being an outdated OS; it continues to evolve and adapt to modern technology trends with different architectures – specifically Alpha (another processor architecture developed by DEC) and Integrity (Intel's Itanium architecture) so, keeping "VAX/" in the name no longer made sense. Since the OS was no longer exclusively for VAX systems. Therefore, the specific hardware prefix (VAX) was removed and was replaced by “open”.

The "Open" prefix provided that the OpenVMS was no longer limited to single hardware architecture (VAX) but supported different hardware platforms like Alpha and Integrity (and now, in the form of the Community License, x86-64 via virtualization). This "openness" in terms of hardware and standards aimed to make the operating system more versatile and adaptable in a changing computing landscape.

The computers that OpenVMS used to run on (VAX and Alpha) are old and no longer made. However, the software that was written for OpenVMS is still very important and used every day by many companies for their key business activities. This is because these applications are often stable, reliable, and tailored to specific needs, making them difficult and costly to replace, even if the underlying hardware is aging.

Motivation for OpenVMS Development

The reason OpenVMS (originally called VMS) was created was to build a **strong, reliable, and flexible operating system** that could make the most of **Digital Equipment Corporation's (DEC)** new computer system, the **VAX**, which came out in the late 1970s. DEC wanted an OS that could handle many users, run multiple programs at once, and be dependable for important business tasks.

Here's a breakdown of the main goals behind creating OpenVMS:

1. **Better Memory Use**

   The VAX's virtual memory capability allowed for running larger programs. To fully utilize this feature and deliver better performance to users, they needed an **operating system** that was specifically designed to manage the virtual memory effectively.

   The VMS operating system was then created, in part, to take advantage of the VAX architecture's virtual memory capabilities. By intelligently managing the swapping of data between RAM and the hard drive, VMS could optimize the use of available memory and allow for more efficient execution of larger and more demanding programs.

2. **Support for Many Users and Programs**

   In the early days of computing, resources were expensive. It was more cost-effective and efficient for multiple individuals to share a single powerful computer rather than each having their own. Businesses and research institutions required systems that could support concurrent access from different terminals or workstations, allowing multiple users to work simultaneously. OpenVMS was built to handle both of these things well.

3. **High Reliability**

   Some industries—like banking and telecommunications—needed systems that almost never crash. They need systems that have **high availability** and **fault tolerance.** These industries rely on their computer systems to be operational continuously, often 24 hours a day, 7 days a week. System crashes or downtime in these areas can lead to significant financial losses, disruptions in essential services, and damage to reputation.

   OpenVMS was designed with features and an architecture aimed at minimizing the occurrence of system failures or crashes. For operations where it's absolutely vital that the system doesn't go down. OpenVMS was an ideal solution.

4. **Strong Security**

   Security was a significant consideration in the development of OpenVMS. The operating system came equipped with a set of powerful features specifically designed to protect the system and its data such as passwords, user access controls, and system logging. By implementing passwords for authentication, controlling who can access what through user access controls, and monitoring system activity with logging, OpenVMS aimed to prevent individuals who were not permitted from viewing, modifying, or otherwise interacting with sensitive data stored on the system.

5. **Designed for the Future**

   DEC strategically designed OpenVMS with future hardware in mind. It designed OpenVMS with an architecture that wasn't tightly bound to the specifics of the initial VAX hardware. They aimed to make it easier to adapt and run the operating system on different underlying hardware platforms that might emerge in the future.

   That’s why OpenVMS could eventually run not just on VAX, over time, it was successfully adapted (ported) to run on:

- **Alpha:** Another processor architecture developed by DEC itself.
- **Itanium:** Intel's 64-bit architecture.
- **Modern x86-64 machines:** The dominant processor architecture in today's desktop, laptop, and server markets (as seen with the current Community License).

Objectives of OpenVMS Operating System

The development of OpenVMS (originally VMS) by Digital Equipment Corporation (DEC) was driven by the need for a secure, stable, and high-performance operating system that could fully utilize the capabilities of the VAX computer architecture. Designed with enterprise and mission-critical applications in mind, OpenVMS has evolved over decades to support a wide range of hardware platforms and to adapt to changing technological landscapes. The system continues to be used in industries where uptime, security, and reliability are non-negotiable.

1. **Reliability and Continuous Operation** One of the primary objectives of OpenVMS was to provide a highly reliable and stable operating environment. This was particularly important for industries such as banking, healthcare, and telecommunications, where downtime can have serious consequences.

- **Fault Tolerance and Clustering** OpenVMS supports clustering technology, allowing multiple systems to work together and share workloads. This helps prevent service interruptions and ensures continuous operation even if one system fails.
- **Data Integrity** The system includes features for ensuring data consistency and protecting against data loss during unexpected shutdowns or hardware failures.

2. **Multi-User and Multi-Tasking Environment** OpenVMS was designed to handle multiple users and processes simultaneously.

- **Efficient Resource Sharing** the OS allows many users to access system resources at the same time without interfering with each other.
- **Advanced Process Scheduling** OpenVMS includes a sophisticated scheduler to manage multiple tasks and optimize system performance under heavy workloads.

3. **Security and Access Control** Given its use in environments handling sensitive information, OpenVMS was built with strong security measures.

- **User Authentication and Authorization** The OS includes robust login systems, password controls, and access permissions.
- **Auditing and Monitoring** OpenVMS can track system activity to detect unauthorized access and assist with compliance requirements.

4. **Virtual Memory Management** To support complex and memory-intensive applications, OpenVMS was developed to make effective use of virtual memory.

- **Support for Large Applications** By using disk space as additional memory, OpenVMS allows larger programs to run efficiently.
- **Memory Protection** Virtual memory management also provides isolation between processes, increasing system stability.

5. **Hardware Compatibility and Portability** Another key objective was to create an operating system that could adapt to new hardware platforms over time.

- **Support for Evolving Architectures** Originally built for VAX, OpenVMS has been successfully ported to Alpha, Itanium, and x86-64 architectures.
- **Longevity and Investment Protection** This hardware flexibility allows organizations to continue using OpenVMS even as they upgrade their infrastructure.

6. **Manageability and Automation** OpenVMS was also designed to be easy to manage and automate for system administrators.

- **Command Line Tools and Scripting** The Digital Command Language (DCL) provides powerful tools for automating tasks and managing system settings.
- **System Monitoring and Maintenance** Built-in utilities help monitor performance and assist in proactive system maintenance.

7. **Community and Ongoing Development** Even after decades, OpenVMS continues to receive updates and has an active user base.

- **Support from VMS Software Inc. (VSI)** VSI continues to develop and maintain OpenVMS, including support for cloud integration and modern hardware.

- **Active User Community** A dedicated community provides support, documentation, and knowledge sharing; ensuring OpenVMS remains a practical choice for critical systems.

REQUIREMENTS

I. Hardware Requirements

OpenVMS was designed with enterprise-level robustness in mind and optimized for stability, scalability, and performance on specific hardware platforms over the decades. It originally targeted VAX systems and later expanded to Alpha, Itanium, and most recently, x86-64 platforms.

A. Supported Architectures

OpenVMS is known for its strong support across multiple generations of DEC and HPE systems. As of recent versions:

1. **VAX Architecture**
   - OpenVMS was originally developed for Digital Equipment Corporation's VAX (Virtual Address eXtension) systems.
   - These systems supported 32-bit virtual memory and were popular in the 1980s and 1990s in enterprise and research institutions.
2. **Alpha Architecture**
   - OpenVMS was ported to the 64-bit Alpha architecture in the early 1990s.
   - Alpha systems delivered enhanced floating-point performance and were often used in high-performance computing, banking, and industrial automation.
3. **Itanium (IA-64)**
   - HPE's Integrity servers used Intel's Itanium processors, and OpenVMS was ported to this platform in the early 2000s.
   - It supported enterprise environments with high availability and mission-critical workloads.
4. **x86-64 (Intel/AMD 64-bit)**
   - The latest port of OpenVMS supports modern x86-64 systems, enabling the OS to run on widely available hardware including virtual machines, desktops, and servers.
   - This port significantly broadened the accessibility of OpenVMS for both legacy users and new deployments.

B. Processor Requirements

1. **Minimum Processor Requirements**
   - VAX systems: Required original DEC VAX processors (32-bit).
   - Alpha systems: Required at least an Alpha EV4 or newer processor.
   - Itanium: Required HP Integrity Itanium 2 processors.
   - x86-64: Requires a 64-bit capable Intel or AMD processor that supports standard virtualization extensions (e.g., VT-x, AMD-V).

2. **Recommended Processor Specifications**
    - Multi-core x86-64 processors (Intel Xeon, AMD EPYC) for modern performance.
    - Virtualization support is beneficial for testing or cloud deployments.

C. Memory Requirements

1. **Minimum Memory**
    - VAX: Minimum of 4 MB RAM (historically).
    - Alpha/Itanium: Typically required 128 MB or more.
    - x86-64: Minimum of 512 MB to 1 GB RAM for basic installations.
2. **Recommended Memory**
    - 2 GB or more recommended for smooth operation on x86-64, especially for multi-user or database workloads.
    - Systems using features like clustering, database management, or large-scale file systems benefit from 4 GB+ RAM.

D. Storage Requirements

1. **Minimum Disk Space**
    - OpenVMS requires at least 2–4 GB of disk space for a minimal install.
    - Base install includes system kernel, utilities, documentation, and basic services.
2. **Recommended Disk Space**
    - 20 GB or more for production systems using layered products (like Oracle Rdb, DECnet, or clustering).
    - SSDs or fast RAID storage recommended for high-performance environments.

E. Peripheral and Network Support

1. **Network Interface Cards (NICs)**
    - Support for a variety of Ethernet controllers.
    - Recent x86-64 versions include drivers for common Intel and Broadcom NICs.
2. **Input/Output Devices**
    - Standard keyboard, mouse, terminal consoles, and serial ports supported.
    - Virtual console and serial console interfaces available for headless or server installs.

II. Software Requirements

OpenVMS includes a powerful suite of built-in tools and utilities tailored for enterprise needs. It also supports many layered software products and services.

A. Operating System Software

1. **Core OS**
    - OpenVMS includes a monolithic kernel with modular components.

    - Features include a file system (ODS-2/ODS-5), DCL (command language), and a powerful job/process control system.
2. **System Utilities**
    - Includes editors, compilers, system management tools, and development libraries.
    - Full support for batch jobs, system scheduling, printing, backup, and clustering.

B. Layered Software Products and Add-ons

1. **DECnet and TCP/IP Services**
    - Native support for DECnet Phase IV and TCP/IP.
    - Network communication, remote access, and file sharing are integral parts of the OS.
2. **Database and Enterprise Software**
    - Supports Oracle Rdb, Oracle DBMS, and other databases.
    - RMS (Record Management Services) provides native indexed file support for transactional apps.
3. **Programming Languages**
    - Native support or compilers available for:
        - C, C++, Fortran, Pascal, BASIC, COBOL, and more.
    - IDEs and build tools are available, and scripting via DCL or foreign command interfaces is supported.
4. **Security and Access Control Software**
    - Integrated security model with user accounts, privileges, ACLs (Access Control Lists), and auditing.
    - Optional encryption libraries and multi-factor authentication add-ons available.
5. **Modern Compatibility**
    - OpenVMS on x86-64 supports running in virtualized environments (e.g., VMware, KVM).
    - Support for SSH, HTTPS, and scripting with modern tools like Python (via ports).

INSTALLATION STEPS:

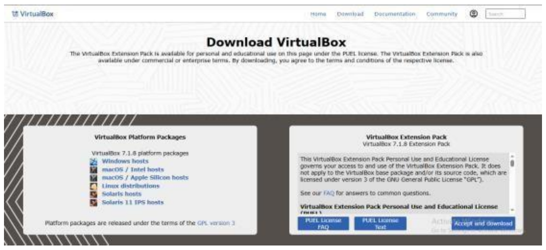
Step 1: Download the Oracle VirtualBox Installer

The first step is to obtain a verified, official copy of the Oracle VirtualBox installation software.

1. Open a web browser and navigate to the official VirtualBox downloads page: "https://www.virtualbox.org/wiki/Downloads".

2. Locate the section for VirtualBox platform packages. Choose the installer corresponding to the Host Operating System, which in this case is "Windows hosts".

3. Click on the link for "Windows hosts". The download of the VirtualBox installer will begin immediately.

1. 

VirtualBox

Home Download Documentation Community

Download VirtualBox

The VirtualBox Extension Pack is available for personal and educational use on this page under the PUEL license. The VirtualBox Extension Pack is also available under commercial or enterprise terms. By downloading, you agree to the terms and conditions of the respective license.

VirtualBox Platform Packages

VirtualBox 7.1.8 platform packages
- Windows hosts
- macOS / Intel hosts
- macOS / Apple Silicon hosts
- Linux distributions
- Solaris hosts
- Solaris 11 IPS hosts

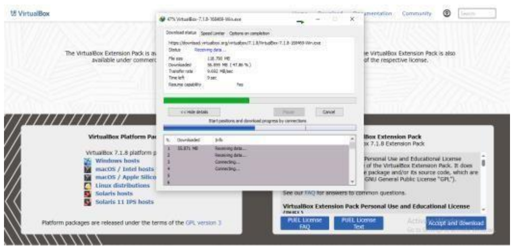
Platform packages are released under the terms of the GPL version 3

VirtualBox Extension Pack
VirtualBox 7.1.8 Extension Pack

This VirtualBox Extension Pack Personal Use and Educational License governs your access to and use of the VirtualBox Extension Pack. It does not apply to the VirtualBox base package and/or its source code, which are licensed under version 3 of the GNU General Public License "GPL").

See our FAQ for answers to common questions.

VirtualBox Extension Pack Personal Use and Educational License (PUEL)

2. 

47% VirtualBox-7.1.8-168469-Win.exe

https://download.virtualbox.org/virtualbox/7.1.8/VirtualBox-7.1.8-168469-Win.exe

Status: Receiving data...

File size: 118.750 MB
Downloaded: 56.899 MB (47.86 %)
Transfer rate: 9.032 MB/sec
Time left: 9 sec
Resume capability: Yes

Start positions and download progress by connections

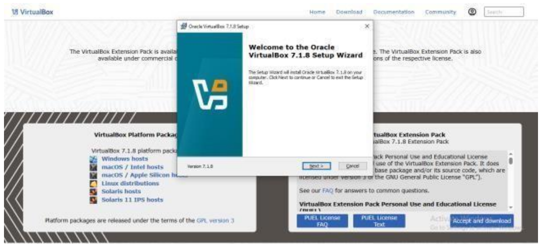
```
N.  Downloaded   Info
1   55.871 MB    Receiving data...
2                Receiving data...
3                Connecting...
4                Connecting...
5
6
```

Step 2: Install Oracle VirtualBox

Once the download is complete, proceed with the installation process.

1. Locate the downloaded VirtualBox installer file (it will likely be an .exe file).
2. Double-click the installer file to begin the installation.
3. Follow the on-screen prompts. When presented with installation options and extra questions, it is generally safe to select "Yes" or "Next" to accept the default configurations for a standard installation.
4. Allow the installation process to complete.

1.



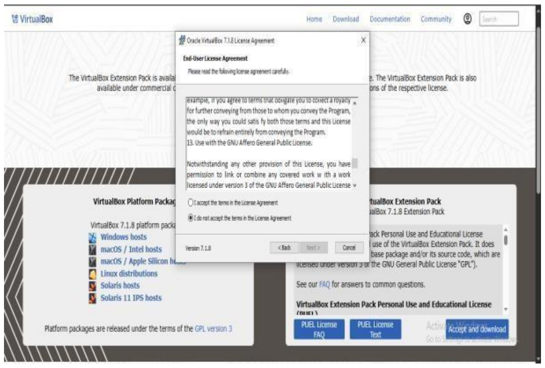
Oracle VirtualBox 7.1.8 Setup

Welcome to the Oracle VirtualBox 7.1.8 Setup Wizard

The Setup Wizard will install Oracle VirtualBox 7.1.8 on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

Version 7.1.8

2.



Oracle VirtualBox 7.1.8 License Agreement

End-User License Agreement

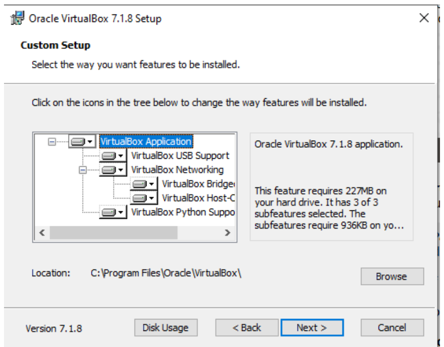
Please read the following license agreement carefully.

example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.
13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License

I accept the terms in the License Agreement [✓ I do not accept the terms in the License Agreement]

Version 7.1.8



Oracle VirtualBox 7.1.8 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 Bridge
    - VirtualBox Host-C
  - VirtualBox Python Suppo

Oracle VirtualBox 7.1.8 application.

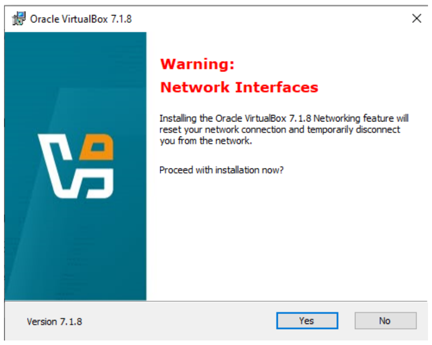
This feature requires 227MB on your hard drive. It has 3 of 3 subfeatures selected. The subfeatures require 936KB on yo...

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

Version 7.1.8

3.

4.


Oracle VirtualBox 7.1.8

Warning:
Network Interfaces

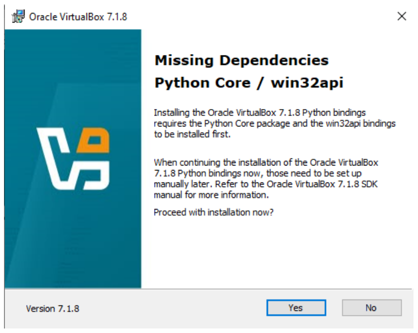
Installing the Oracle VirtualBox 7.1.8 Networking feature will reset your network connection and temporarily disconnect you from the network.

Proceed with installation now?

Version 7.1.8

Yes No

5.


Oracle VirtualBox 7.1.8

Missing Dependencies
Python Core / win32api

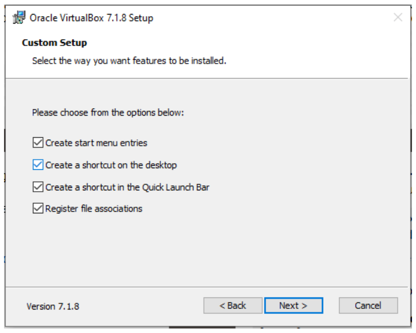
Installing the Oracle VirtualBox 7.1.8 Python bindings requires the Python Core package and the win32api bindings to be installed first.

When continuing the installation of the Oracle VirtualBox 7.1.8 Python bindings now, those need to be set up manually later. Refer to the Oracle VirtualBox 7.1.8 SDK manual for more information.

Proceed with installation now?

Version 7.1.8

Yes No


Oracle VirtualBox 7.1.8 Setup

Custom Setup

Select the way you want features to be installed.

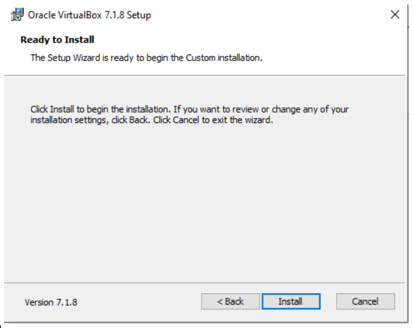
Please choose from the options below:

- ☑ Create start menu entries
- ☑ Create a shortcut on the desktop
- ☑ Create a shortcut in the Quick Launch Bar
- ☑ Register file associations

Version 7.1.8

< Back Next > Cancel

5. 7.


Oracle VirtualBox 7.1.8 Setup

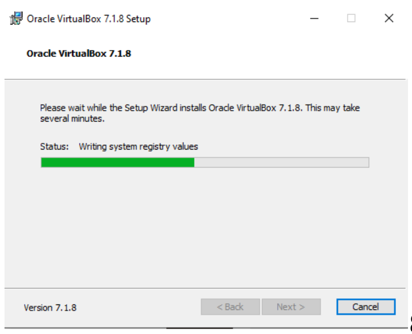
Ready to Install

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 7.1.8

< Back Install Cancel


Oracle VirtualBox 7.1.8 Setup

Oracle VirtualBox 7.1.8

Please wait while the Setup Wizard installs Oracle VirtualBox 7.1.8. This may take several minutes.

Status: Writing system registry values

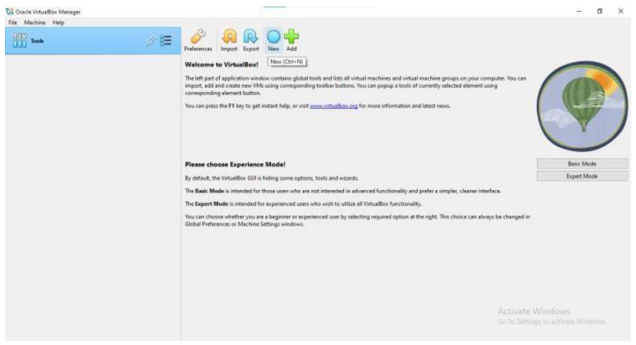
Version 7.1.8

< Back Next > Cancel

8.

Step 3: Create a New Virtual Machine for the OS

1. Open the Oracle VirtualBox application.
2. Click on the "New" icon located on the left side of the toolbar (it's usually the first icon). This will open the "Create Virtual Machine" wizard.



Welcome to VirtualBox!

The left part of application window contains global tools and lists all virtual machines and virtual machine groups on your computer. You can import, add and create new VMs using corresponding toolbar buttons. You can popup a tools of currently selected element using corresponding element button.

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

Please choose Experience Mode!

By default, the VirtualBox GUI is hiding some options, tools and wizards.

The **Basic Mode** is intended for those users who are not interested in advanced functionality and prefer a simpler, cleaner interface.

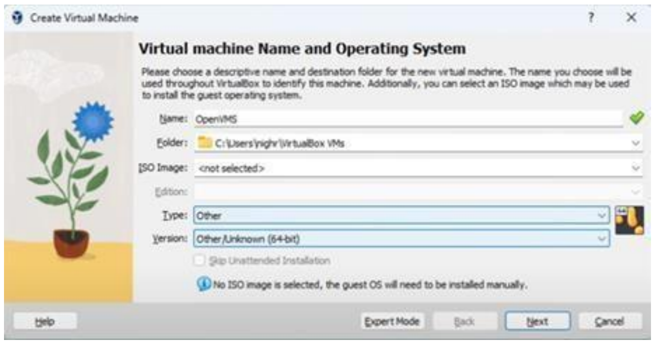
The **Expert Mode** is intended for experienced users who wish to utilize all VirtualBox functionality.

You can choose whether you are a beginner or experienced user by selecting required option at the right. This choice can always be changed in Global Preferences or Machine Settings windows.

Activate Windows
Go to Settings to activate Windows.

Step 4: Configure Basic Virtual Machine Settings

1. **Name:** In the "Name" field, enter any name. I picked "OpenVMS" to identify the virtual machine.
2. **ISO Image:** Leave the "ISO Image" field as "Not selected" for now. It will be specified in the installation media later.
3. **Type:** In the "Type" dropdown menu, select "Other".
4. **Version:** In the "Version" dropdown menu, select "Other/Unknown".
5. Click "Next" to proceed.



Virtual machine Name and Operating System

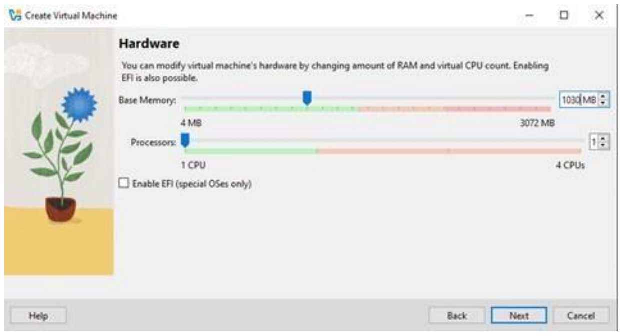
Please choose a descriptive name and destination folder for the new virtual machine. The name you choose will be used throughout VirtualBox to identify this machine. Additionally, you can select an ISO image which may be used to install the guest operating system.

Name: OpenVMS
Folder: C:\Users\nighr\VirtualBox VMs
ISO Image: <not selected>
Edition:
Type: Other
Version: Other/Unknown (64-bit)
☐ Skip Unattended Installation

No ISO image is selected, the guest OS will need to be installed manually.

Step 5: Configure Memory and CPU

1. **Memory Size:** Set the "Memory size" to "1030 MB". You can either type this value or use the slider.
2. **Processors:** Set the "Processors" to "1".
3. **Enable EFI:** Ensure the "Enable EFI (special OSes only)" checkbox is **unchecked**. This is because the target operating system requires a traditional BIOS environment, and enabling EFI might prevent it from booting correctly.
4. Click "Next" to continue with further virtual machine setup.


Create Virtual Machine

Hardware

You can modify virtual machine's hardware by changing amount of RAM and virtual CPU count. Enabling EFI is also possible.

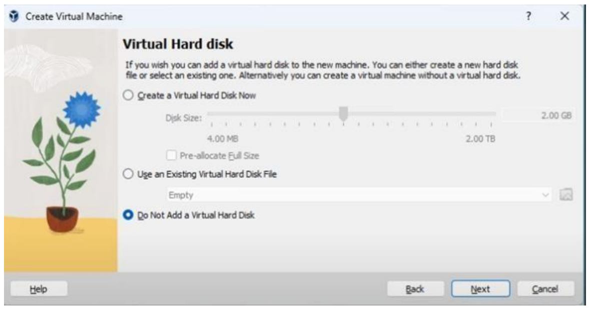
Base Memory: 1030 MB (4 MB – 3072 MB)

Processors: 1 (1 CPU – 4 CPUs)

☐ Enable EFI (special OSes only)

Step 6: Configure Hard Disk Allocation

1. In the "Virtual Hard Disk" window, select the option "Do not add a virtual hard disk" for now.
2. Click "Next".


Create Virtual Machine

Virtual 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 an existing one. Alternatively you can create a virtual machine without a virtual hard disk.

○ Create a Virtual Hard Disk Now

Disk Size: 2.00 GB (4.00 MB – 2.00 TB)

☐ Pre-allocate Full Size

○ Use an Existing Virtual Hard Disk File

Empty

● Do Not Add a Virtual Hard Disk

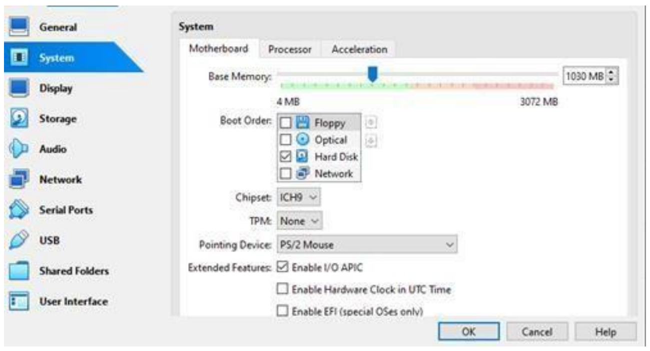
Step 7: Review Virtual Machine Summary

1. The wizard will now display a summary of the configuration settings you have chosen for the "OpenVMS" virtual machine.
2. Review these settings to ensure they are as intended.

3. Click "Finish". VirtualBox will now create the virtual machine based on these specifications.

Step 8: Adjust Virtual Machine System Settings

1. In the VirtualBox Manager, select the newly created "OpenVMS" virtual machine.
2. Click on the "Settings" icon in the toolbar.
3. In the "Settings" window, navigate to the "System" tab.
4. In the "Boot Order" section, uncheck the boxes for "Floppy", "Optical", and "Network". Ensure that the boot order prioritizes the hard disk (which will be configured later with the ISO).
5. Set the "Chipset" to "ICH9".
6. Set "TPM" to "None".
7. Navigate to the "Pointing Device" dropdown and select "PS/2 Mouse".
8. Ensure the "Enable I/O APIC" checkbox is checked.
9. Click "OK" to save these system settings.



Base Memory: 1030 MB (4 MB – 3072 MB)

Boot Order:
- ☐ Floppy
- ☐ Optical
- ☑ Hard Disk
- ☐ Network

Chipset: ICH9

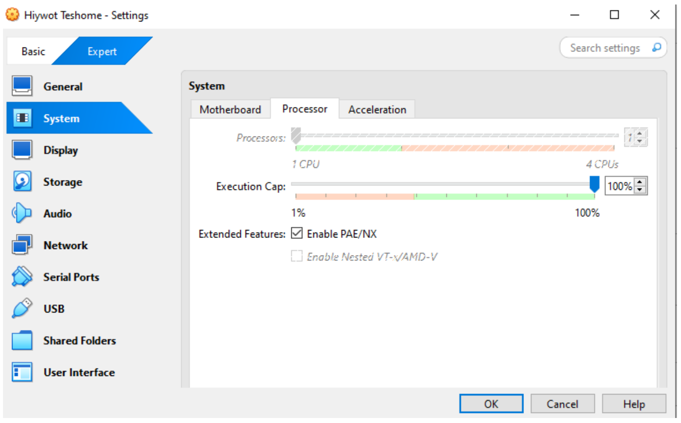
TPM: None

Pointing Device: PS/2 Mouse

Extended Features:
- ☑ Enable I/O APIC
- ☐ Enable Hardware Clock in UTC Time
- ☐ Enable EFI (special OSes only)

Step 9: Verify Processor Settings

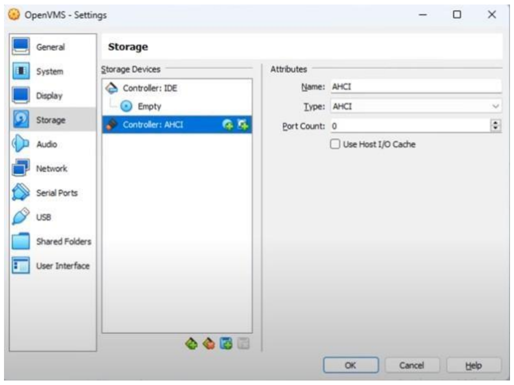
1. While still in the "Settings" window for the "OpenVMS" virtual machine, navigate to the "Processor" tab.
2. Confirm that the "Processor(s)" are set to "1", as configured earlier.
3. Click "OK" if you made any changes, otherwise, you can proceed.



Processors: 1
1 CPU 4 CPUs
Execution Cap: 100%
1% 100%
Extended Features: ☑ Enable PAE/NX
☐ Enable Nested VT-x/AMD-V

Step 10: Add an AHCI Storage Controller

1. In the "Settings" window, navigate to the "Storage" tab.
2. Click on the "+" icon with the tooltip "Adds new storage controller" located below the list of "Devices".
3. Select "Add AHCI Controller". A new controller will appear in the storage tree.

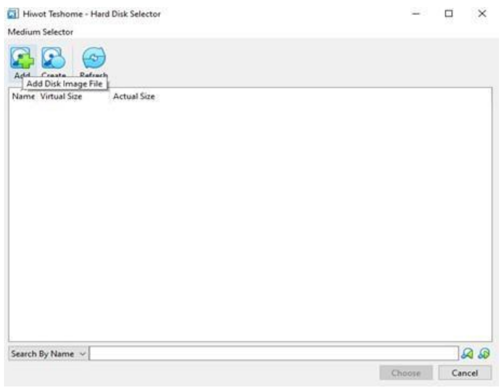


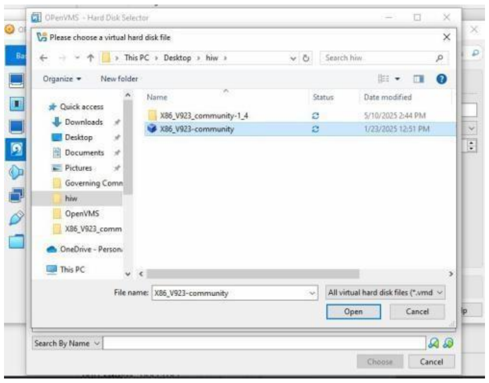
Name: AHCI
Type: AHCI
Port Count: 0
☐ Use Host I/O Cache

Step 11: Add the OpenVMS ISO Image

1. Click on the second "+" icon (with a CD/DVD symbol and the tooltip "Adds optical drive") under the newly added AHCI Controller.
2. In the "Choose Virtual Optical Disk File" window, click on the "Add" button (usually a "+" icon).
3. Navigate to the location where you saved the OpenVMS ISO image obtained from VMS Software, Inc. (VSI).
4. Select the ISO file and click "Open".
5. Back in the "Choose Virtual Optical Disk File" window, select the added ISO image.

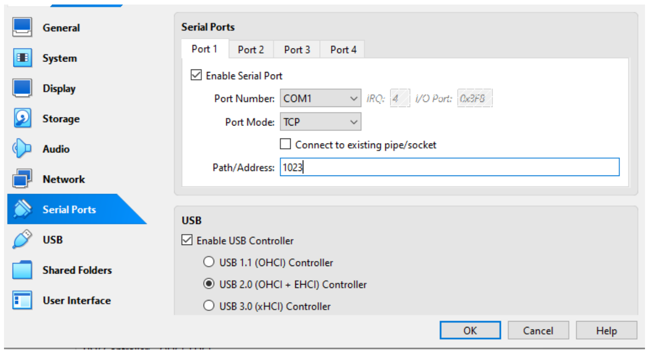
6. Click "Choose". The OpenVMS ISO image is now associated with the virtual optical drive.





Step 12: Enable and Configure Serial Port

1. In the "Settings" window, navigate to the "Serial Ports" tab.
2. Ensure the "Enable Serial Port" checkbox is checked.
3. **Port Number:** Set the "Port Number" to "COM1".
4. **Port Mode:** Set the "Port Mode" to "TCP".
5. Ensure the "Connect to existing pipe/socket" checkbox is **unchecked**.
6. **Path/Address:** Set the "Path/Address" to "1023".
7. Click "OK" to save the serial port configuration.



Serial Ports

☑ Enable Serial Port

Port Number: COM1 IRQ: 4 I/O Port: 0x3F8

Port Mode: TCP

☐ Connect to existing pipe/socket

Path/Address: 1023

USB

☑ Enable USB Controller

○ USB 1.1 (OHCI) Controller

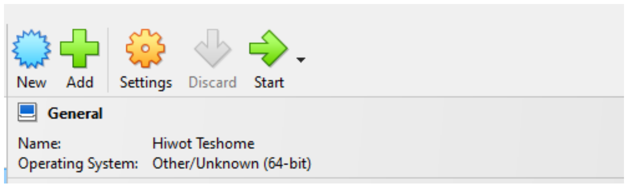
● USB 2.0 (OHCI + EHCI) Controller

○ USB 3.0 (xHCI) Controller

Step 13: Start the Virtual Machine Installation

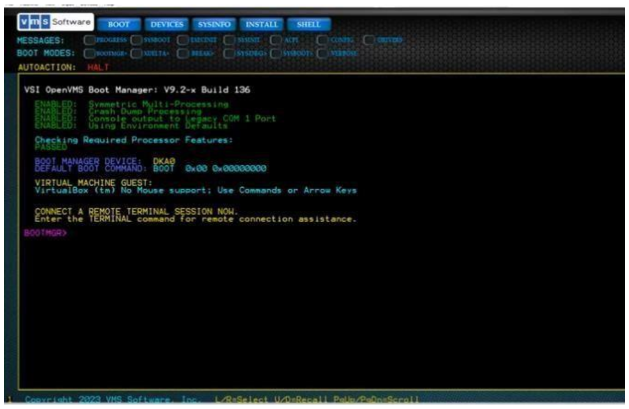
1. In the VirtualBox Manager, ensure the "OpenVMS" virtual machine is selected.

2. Click the "Start" button (usually a green arrow) in the toolbar. This will power on the virtual machine and begin the installation process using the OpenVMS ISO image you attached.



Name: Hiwot Teshome
Operating System: Other/Unknown (64-bit)

This is what the OpenVMS was supposed to like when it is installed:

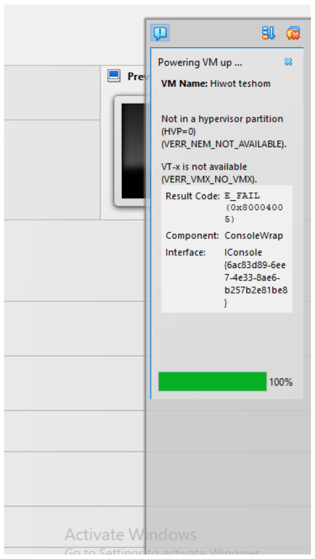


```
MESSAGES:
BOOT MODES:
AUTOACTION:  HALT

VSI OpenVMS Boot Manager: V9.2-x Build 136
   ENABLED:   Symmetric Multi-Processing
   ENABLED:   Crash Dump Processing
   ENABLED:   Console output to Legacy COM 1 Port
   ENABLED:   Using Environment Defaults

   Checking Required Processor Features:
   PASSED

   BOOT MANAGER DEVICE:   DKA0
   DEFAULT BOOT COMMAND:  BOOT   0x00 0x00000000

   VIRTUAL MACHINE GUEST:
   VirtualBox (tm) No Mouse support; Use Commands or Arrow Keys

   CONNECT A REMOTE TERMINAL SESSION NOW.
   Enter the TERMINAL command for remote connection assistance.
BOOTMGR>
```

When I tried to start the virtual machine set up for OpenVMS in VirtualBox, I ran into a serious error that stopped it from running. The VirtualBox displayed the following error messages:

1. **Not in a hypervisor partition (HVP=0) (VERR_NEM_NOT_AVAILABLE):** This error strongly suggests that the necessary hypervisor environment for running the virtual machine is not available or properly configured on my host system. A hypervisor is the underlying software layer that manages virtual machines. This error often arises when hardware virtualization features are not enabled or are being interfered with.

2. **VT-x is not available (VERR_VMX_NO_VMX):** This is a more specific indicator that Intel Virtualization Technology (VT-x) or AMD-V (the equivalent for AMD processors) is either not supported by my CPU, not enabled in the system's BIOS/UEFI settings, or is being used exclusively by another hypervisor. These CPU virtualization extensions are essential for the efficient operation of most modern virtualization software like VirtualBox. This was the reply that I was receiving:



```
Powering VM up ...
VM Name: Hiwot teshom

Not in a hypervisor partition
(HVP=0)
(VERR_NEM_NOT_AVAILABLE).

VT-x is not available
(VERR_VMX_NO_VMX).
Result Code: E_FAIL
             (0x8000400
             5)
Component:   ConsoleWrap
Interface:   IConsole
             {6ac83d89-6ee
             7-4e33-8ae6-
             b257b2e81be8
             }
100%
```

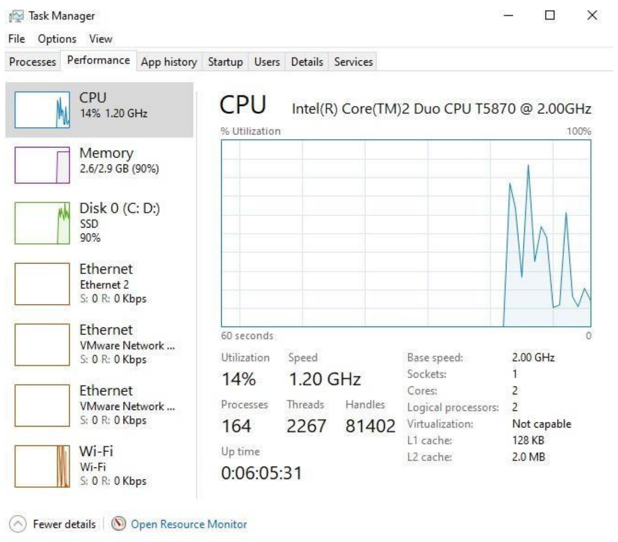
Troubleshooting Steps and Analysis:

To address these errors, I undertook the following troubleshooting steps:

Verification of CPU Virtualization Support: I checked the specifications of my computer's CPU to confirm if it supports Intel VT-x or AMD-V. I checked the 'Virtualization' status in the Performance tab of the Task Manager. Instead of 'Enabled' or 'Disabled', the status showed 'Not applicable'.

The primary reason why the virtualization says "Not applicable" in the Task Manager is that my PC’s processor is an Intel(R) Core(TM)2 Duo CPU T5870 @ 2.00GHz. While many Core 2 Duo processors did support Intel Virtualization Technology (VT-x), not all models did. Specifically, based on Intel's official specifications and third-party sources, the Intel Core 2 Duo T5870 does NOT support Intel VT-x.

The second reason is that the BIOS version is TOSHIBA V1.20, dated 10/28/2009. This is also quite old BIOS. Even if the CPU did support virtualization, older BIOS might not have the necessary settings or support for enabling it correctly.

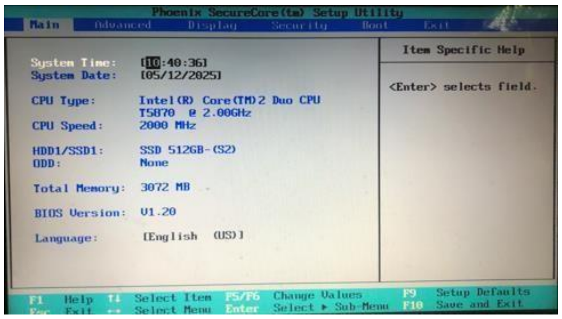


```
CPU                 14%  1.20 GHz
Memory              2.6/2.9 GB (90%)
Disk 0 (C: D:)      SSD 90%
Ethernet            Ethernet 2            S: 0 R: 0 Kbps
Ethernet            VMware Network ...    S: 0 R: 0 Kbps
Ethernet            VMware Network ...    S: 0 R: 0 Kbps
Wi-Fi               Wi-Fi                 S: 0 R: 0 Kbps

CPU   Intel(R) Core(TM)2 Duo CPU T5870 @ 2.00GHz
% Utilization

Utilization   Speed          Base speed:          2.00 GHz
14%           1.20 GHz       Sockets:             1
                             Cores:               2
Processes   Threads  Handles Logical processors:  2
164         2267     81402   Virtualization:      Not capable
                             L1 cache:            128 KB
Up time                      L2 cache:            2.0 MB
0:06:05:31
```

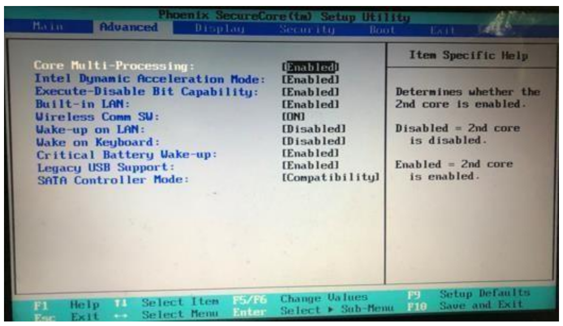
BIOS/UEFI Settings Check: I accessed the system's BIOS/UEFI settings during startup to ensure that the virtualization technology (VT-x/AMD-V) was enabled.

To ensure that hardware virtualization was enabled at the firmware level, I restarted my computer and pressed the F2 key repeatedly during startup to enter the BIOS/UEFI setup utility.

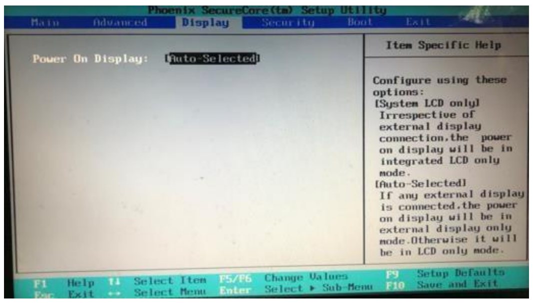
Once in the BIOS/UEFI, I navigated through the menus using the arrow keys to find the settings related to CPU configuration or advanced features. But I could not find any options related to virtualization.


Phoenix SecureCore(tm) Setup Utility
Main Advanced Display Security Boot Exit
F1 Help ↑↓ Select Item F5/F6 Change Values F9 Setup Defaults
Esc Exit ↔ Select Menu Enter Select ▸ Sub-Menu F10 Save and Exit

```
System Time:     [10:40:36]               Item Specific Help
System Date:     [05/12/2025]
                                          <Enter> selects field.
CPU Type:        Intel(R) Core(TM)2 Duo CPU
                 T5870  @ 2.00GHz
CPU Speed:       2000 MHz

HDD1/SSD1:       SSD 512GB-(S2)
ODD:             None

Total Memory:    3072 MB

BIOS Version:    V1.20

Language:        [English  (US)]
```


Phoenix SecureCore(tm) Setup Utility
Main Advanced Display Security Boot Exit
F1 Help ↑↓ Select Item F5/F6 Change Values F9 Setup Defaults
Esc Exit ↔ Select Menu Enter Select ▸ Sub-Menu F10 Save and Exit

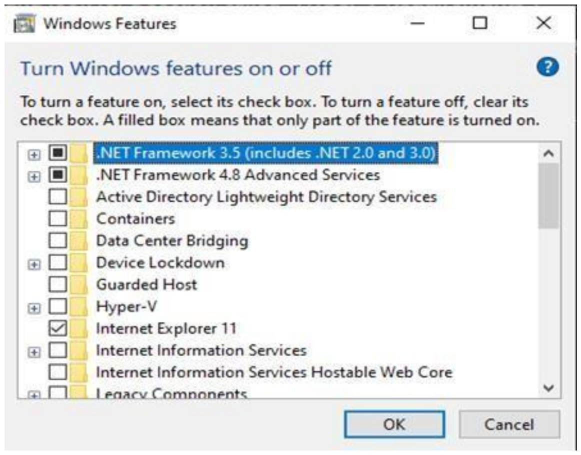
```
Core Multi-Processing:               [Enabled]        Item Specific Help
Intel Dynamic Acceleration Mode:     [Enabled]
Execute-Disable Bit Capability:      [Enabled]        Determines whether the
Built-in LAN:                        [Enabled]        2nd core is enabled.
Wireless Comm SW:                    [ON]
Wake-up on LAN:                      [Disabled]       Disabled = 2nd core
Wake on Keyboard:                    [Disabled]         is disabled.
Critical Battery Wake-up:            [Enabled]
Legacy USB Support:                  [Enabled]        Enabled = 2nd core
SATA Controller Mode:                [Compatibility]    is enabled.
```


Phoenix SecureCore(tm) Setup Utility
Main Advanced Display Security Boot Exit
F1 Help ↑↓ Select Item F5/F6 Change Values F9 Setup Defaults
Esc Exit ↔ Select Menu Enter Select ▸ Sub-Menu F10 Save and Exit

```
Power On Display:  [Auto-Selected]       Item Specific Help

                                         Configure using these
                                         options:
                                         [System LCD only]
                                          Irrespective of
                                          external display
                                          connection,the  power
                                          on display will be in
                                          integrated LCD only
                                          mode.
                                         [Auto-Selected]
                                          If any external display
                                          is connected.the power
                                          on display will be in
                                          external display only
                                          mode.Otherwise it will
                                          be in LCD only mode.
```

This is caused due to the fact that my PC's CPU itself doesn't have the hardware virtualization capabilities, the BIOS settings won't offer an option to enable it.

Investigating Hyper-V (if you are on Windows): I checked if Hyper-V was enabled on my system. If Hyper-V, Microsoft's native hypervisor, is enabled, it can sometimes exclusively grab the virtualization extensions. In this scenario, VirtualBox might not be able to use them, and the Task Manager might report "Not applicable" because the features are in use by another hypervisor. But I made sure to disable the Hyper-V, and it still did not work.


Windows Features

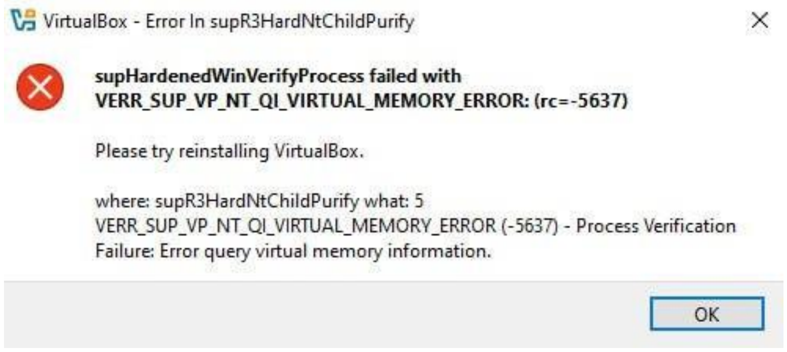
Turn Windows features on or off

To turn a feature on, select its check box. To turn a feature off, clear its check box. A filled box means that only part of the feature is turned on.

- ■ .NET Framework 3.5 (includes .NET 2.0 and 3.0)
- ■ .NET Framework 4.8 Advanced Services
- ☐ Active Directory Lightweight Directory Services
- ☐ Containers
- ☐ Data Center Bridging
- ☐ Device Lockdown
- ☐ Guarded Host
- ☐ Hyper-V
- ☑ Internet Explorer 11
- ☐ Internet Information Services
- ☐ Internet Information Services Hostable Web Core
- ☐ Legacy Components

VirtualBox reinstallation: I considered the possibility of a corrupted VirtualBox installation and attempted a reinstallation of the software. This did not resolve the issue.

On my first attempt, the operating system installation failed, primarily due to issues with my PC. I then tried using a different device to install it, but the problem continued. Each time I started the installation, the same error kept appearing:


VirtualBox - Error In supR3HardNtChildPurify

supHardenedWinVerifyProcess failed with VERR_SUP_VP_NT_QI_VIRTUAL_MEMORY_ERROR: (rc=-5637)

Please try reinstalling VirtualBox.

where: supR3HardNtChildPurify what: 5
VERR_SUP_VP_NT_QI_VIRTUAL_MEMORY_ERROR (-5637) - Process Verification Failure: Error query virtual memory information.

This is a **VirtualBox security-related error** that happens because VirtualBox's "hardening" process (which checks for security and integrity of the environment) fails to verify or access virtual memory correctly on the system.

The Common Causes for this error are:

1. Third-party antivirus or security software interfering with VirtualBox.
2. Windows Defender or firewall blocking parts of VirtualBox.
3. Missing or corrupted VirtualBox installation files.

I tried to reinstall VirtualBox, disable antivirus and firewall temporarily, enable virtualization in BIOS, I made sure windows defender or windows firewall isn't blocking VirtualBox. Unfortunately, even after doing that, the error persisted and I couldn’t get the OS to install.

In summary, despite all the ways I tried to install the operating system I kept getting errors about a missing hypervisor and VT-x not being available, indicating a fundamental issue preventing the successful setup of the OpenVMS virtual machine on my current system.

Issues faced while installation

- **Initial delay in receiving the confirmation email:** I tried to download the OpenVMS OS from the official website which was the VMS Software, Inc. (VSI). The link was: https://vmssoftware.com/community/community-license/. But in order to get information on how to download the OS and to get a license I filled out a form on the official website but it took a very long time.

This led me to wait several days for it and wasted my time.

- A lack of readily available and comprehensive documentation and tutorials. This led me to struggle when downloading the OS.
- The main problem I faced and the reason why I did not install the operating system is that the CPU installed on my PC does not have a built in hardware capabilities for virtualization.

Solutions for problems

- For the email since there was no specific action was required other than waiting and checking the spam folder, I did not do anything other than that.
- Since there was shortage of resources I tried to find other ways and sites to get more information regarding the operating system.
- Since the virtualization was at first disabled I tried to enable it through the BIOS but I was not able to since it was not applicable.

File system supported by OpenVMS

Over many years, as technology has changed and improved, OpenVMS has been updated to work with different types of file systems. A **file system** is the method an operating system uses to arrange and manage files on a storage device, like a hard drive or CD-ROM. In OpenVMS, these file systems are known as **ODS formats**, which stands for **On-Disk Structure**. Each ODS format describes a specific way that information is stored, organized, and read on a disk. OpenVMS supports severalof these formats to make sure it can handle both older and newer storage technologies efficiently. There are five levels of

ODS:

ODS-1

- This is the **oldest** format.
- It is used by **RSX-11**, an older operating system.
- It's a **flat** file system. meaning that it is a structure or organization with **few or no hierarchical levels**
- VMS could read it for compatibility, but **VMS itself didn't use it**. When VMS came out, many users still had disks formatted with **RSX-11's file system**, specifically **ODS-1**. VMS couldn't just ignore all those ODS-1 disks. So it had to be **able to read them** so, users could access or migrate their files. Even though VMS could read ODS-1, it **didn't use it** for its own purposes. Instead, VMS used **ODS-2**.
- Replaced by more modern systems like ODS-2 and ODS-5.

ODS-2

- This is the **default and most widely used** file system on OpenVMS. Key characteristics include:

- Hierarchical directory structure.
- Case-insensitive filenames (uppercase alphanumeric plus hyphen, dollar sign, and underscore).
- Filename length limited to 39 characters for the name and 39 for the type.
- Supports file versioning (up to 32,767 versions).
- Supports access control lists (ACLs) for security.
- System disks in an OpenVMS environment **must** be ODS-2.
- Compatible with both VAX and Alpha (and Integrity) architectures for data sharing.

ODS-3 and ODS-4

- These are **special cases** for **CD-ROMs**.
- ODS-3 supports **ISO 9660**, the standard for most CD-ROMs.
- ODS-4 supports **High Sierra Format**, an older CD-ROM format.
- These aren't commonly talked about by their ODS names.

ODS-5: This is a **superset of ODS-2**, introduced to better interoperate with systems like Windows NT. Its key features include:

- Case-preserving but generally case-insensitive behavior (the case of the filename is stored but not typically used for comparisons unless applications are specifically designed to be case-sensitive).
- Longer filenames (up to 238 bytes, allowing for more characters and Unicode support).
- Deeper directory structures are supported compared to the practical limitations sometimes encountered with older software on ODS-2.
- Supports features like hard links (though this support came later with specific OpenVMS versions to meet POSIX requirements).

- ODS-5 volumes can only be fully accessed by OpenVMS Alpha (and Integrity) systems running version 7.2 or later. VAX systems running later versions can see ODS-2 style names on ODS-5 disks but not the extended names.

Advantages and disadvantages of OpenVMS

I. Advantages

Process Isolation

OpenVMS keeps each running program (called a "process") separate from the others. This means if one program crashes or makes a mistake, it won't crash the whole system or mess with other programs. This helps keep everything stable and running smoothly.

Clustering

OpenVMS can link multiple computers together so they work as one big, powerful system. This is called **clustering**. If one computer in the cluster has a problem, the others can take over — so our system stays available and keeps running. It's great for businesses that need their systems to be online all the time.

Robust Security

Security is built deep into OpenVMS. It has strong tools for:

- Making sure only the right people can log in (authentication),
- Controlling who can access or change files (access control),
- Keeping a record of who did what on the system (auditing).

All this helps protect against misuse, mistakes, or attacks.

File System Features

The OpenVMS file system comes with powerful features like:

- **File-level security** – controlling access to individual files.
- **File versioning** – keeping older versions of a file when you save changes, in case you need to go back.
- **Support for large files** – can handle big chunks of data with no trouble.

Reliability

OpenVMS is designed to run reliably, even in tough conditions. It:

- Checks for hardware and software problems,
- Can make copies of important system components (mirroring),
- Detects and fixes errors before they cause issues.

This makes it ideal for companies that can’t afford downtime.

Scalability

As your business grows, OpenVMS can grow with you. Whether you're running it on a single machine or across many, it can handle more users, more data, and more tasks without slowing down or crashing.

Networking Capabilities

OpenVMS supports many types of networks and communication tools, so it can easily connect to other systems and share information. It works with a variety of network protocols, making it flexible in different environments.

Compatibility with Older Systems

OpenVMS can still run software made for older versions of the system — and on older hardware too. This makes it perfect for organizations that rely on **legacy applications** but still want to upgrade or modernize without starting from scratch. It supports smooth migrations and easy integration with newer systems.

II. Disadvantages

Limited Hardware Support

- OpenVMS was first created for **DEC** computers (like **VAX** and **Alpha** systems) in the 1980s and 90s. These were **custom-built** computers, not regular PCs or laptops like we use today. Over time, OpenVMS was **adapted** to run on other processors: **Itanium** and **x86-64**.
- It lagged behind modern operating systems. Other systems like **Linux** or **Windows** are constantly updated to support the **latest processors, motherboards, and graphics cards**. OpenVMS doesn’t always support the **newest hardware right away**, and sometimes not at all.
- It is not ideal for general-purpose desktop or laptop usage. We **can’t just install OpenVMS** on your everyday laptop or gaming PC. It’s meant for **servers**, **specialized systems**, or **industrial environments**, not general-purpose use like browsing, gaming, or media.

Limited Software Ecosystem

- OpenVMS **doesn’t have as many apps** or development tools (like compilers, editors, libraries) as Linux or Windows. If you're used to things like **Visual Studio**, **Python packages**, **Node.js**, **Docker**, etc.—you might **not find them easily** or **they might not work** on OpenVMS. The **software ecosystem** (the collection of tools and apps available) is **much smaller** and more outdated.
- New or open-source software often needs **manual porting** to work on VMS.

Vendor Lock-In

- VMS Software Inc. (VSI) has done a solid job modernizing OpenVMS and porting it to new hardware like x86-64. But we're tied to their support and updates.
- Unlike Linux, which is open-source (you can view, change, and share the code freely), OpenVMS is closed-source. This means:
- We can't modify the operating system yourself.
- We can't share or improve the core code.
- We have to wait for VSI to make changes.

This is what we call a **vendor lock-in**—you're dependent on one company for the OS's future.

Complexity

- Very different architecture and commands compared to Unix/Linux/Windows.
- Takes time to learn, especially if you're coming from other OS environments.

DCL (Digital Command Language) is the scripting and command language we use to control OpenVMS from the terminal but, the syntax is old-fashioned and hard to learn for newcomers.

Networking and Internet

- Networking support is solid, but not as modern or flexible as what you'd get with Linux.
- Some newer protocols or web technologies might be unsupported or harder to implement.

Cost

- Commercial licensing – while not outrageous, it's **not free** like Linux.
- Long-term cost can be an issue for smaller businesses or startups.

CONCLUSION

OpenVMS was originally created for DEC's VAX computers, and it's known for being super reliable and long-lasting. From the beginning, it was designed to handle multiple users at once, keep systems secure, stay up and running without interruption, and make smart use of memory—things that are just as important today as they were back then.

Even though it's been around for decades, OpenVMS has kept evolving. It's adapted to new owners, new hardware, and changing technology, all while protecting the time and money companies have put into their software.

It's still used in big industries like banking, healthcare, and telecom because it's very stable, secure, and doesn't crash easily. There's also an active group of users and developers (led by VMS Software Inc.) who keep improving it, making OpenVMS a solid choice for businesses that need their systems to run smoothly.

FUTURE OUTLOOK AND RECOMMENDATIONS:

OpenVMS has a strong track record, but to stay relevant in the future, it needs to keep evolving and improve in some key areas. Here's what looks promising and what needs attention:

Good Signs and Opportunities:

- **Cloud Support**: OpenVMS is starting to work with big cloud platforms like AWS and Azure. This makes it easier to use and scale. It's important to keep pushing this forward.
- **New Hardware Compatibility**: OpenVMS now runs on x86-64 systems, which are cheaper and more common. This opens the door for more users and should be optimized to make it run even better on today's hardware.
- **Active Community**: There's a strong group of users and experts who support each other. Keeping this community active by encouraging collaboration and maybe even open-source projects will help drive innovation and support.
- **Stick to What It Does Best**: OpenVMS is known for being reliable, secure, and good at handling multiple processes safely. It should keep focusing on these strengths, especially for industries where this really matters.
- **Better Tools for Developers**: While old tools still matter, bringing in modern programming tools like Python can attract new developers and make the system easier to work with today.

Challenges and How to Tackle Them:

- **Not Enough Software Options**: The range of software that works with OpenVMS is still limited. VSI (the company behind it) should consider partnerships to bring more open-source tools to the platform and show off successful apps that already run on it.
- **Dependence on One Vendor**: Some people worry about relying only on VSI. Making future plans more transparent or allowing more community involvement could help build trust.
- **Steep Learning Curve**: The system and its scripting language (DCL) can be hard for newcomers. Making training easier, creating better documentation, and possibly building simpler graphical tools could help.
- **Old-School Reputation**: Some people still think OpenVMS is outdated. Highlighting its cloud features, support for modern hardware, and sharing real-life success stories can help change this perception.
- **Modern Networking Support**: To work well with today's tech, OpenVMS needs to keep adding support for current internet and networking standards.

What is Virtualization?

Virtualization is a technology that lets one physical computer act like many smaller computers. It uses special software to split up the computer's hardware (like the CPU, memory, and storage) into virtual machines (VMs). Each VM acts like its own separate computer with its own operating system, even though they all run on the same physical machine.

This makes it easier to save money and use resources more efficiently, since we can run many systems on one device. There are three types of server virtualization, namely as:

Full Virtualization

- It’s like a completely fake computer created inside a real one.
- The **hypervisor** (a special program) tricks the operating system into thinking it’s running on real hardware (like its own CPU, RAM, hard drive, etc.).
- The guest operating system has no idea it’s being faked—it acts like it's on a real physical machine. Examples include VMware or VirtualBox.

Para-Virtualization

- The guest OS knows it’s in a virtual environment.
- It’s modified to work better with the hypervisor by using special communication (called hypercalls).
- Since the OS cooperates with the hypervisor, it's faster than full virtualization.

OS-Level Virtualization

- This doesn’t create fake hardware. Instead, it creates separate spaces (containers) inside the same operating system.
- Each container feels like its own little computer but they all share the same OS kernel underneath.

Why use Virtualization?

Saves Money

Virtualization lets you run many virtual machines (VMs) on one physical computer. This means you don’t need to buy extra hardware, which saves a lot of money. You can also make better use of resources that would otherwise sit idle.

More Flexible & Reliable

VMs aren’t tied to one computer. You can back them up, move them, or restart them quickly if something goes wrong. That means less downtime and faster recovery during problems.

High Availability

If one VM fails, another can take over right away. This keeps your systems running smoothly without interruption.

Easier to Manage

Instead of handling many physical machines, you can manage all your VMs from one place. It's quicker to update, secure, and fix systems. You can even scale easily by adding more VMs.

Great for Developers

Developers can clone VMs to create testing environments without touching the live system. This makes building, testing, and launching new features faster and safer.

Better for the Environment

Fewer physical machines mean less energy use. That’s good for the planet—and helps cut down electricity costs too.

How does Virtualization Works in modern operating systems?

Virtualization in modern operating systems is a powerful technology that allows a single physical machine to run multiple operating systems or instances of the same operating system concurrently. It achieves this by creating virtual machines (VMs). The main process’s that are included in the virtualization process is as follows:

Step 1: Install the Hypervisor

- First, we install a **hypervisor**, which is software that helps create and manage virtual machines (VMs).

Step 2: Create Virtual Machines (VMs)

Using the hypervisor, you set up VMs by choosing:

- **Virtual hardware** – how much CPU, memory (RAM), and storage space you want to give to the VM.
- **Operating system** – what system you want to install inside the VM (like Windows, Linux, or OpenVMS).
- **Virtual hard drive** – a file that acts like a real hard disk for the VM.
- **Network connection** – a virtual network card that lets the VM connect to the internet or local network.

Step 3: Install the Operating System

- Once the VM is ready, you install the OS inside it, just like you would on a real computer.
- The OS runs as if it’s on physical hardware, but it's actually using the virtual hardware created by the hypervisor.

Step 4: Manage Resources

- The hypervisor makes sure each VM gets the right amount of CPU, RAM, and storage.
- It also prevents one VM from using too much and slowing down the others.

- Smart features help balance the resources automatically.

Step 5: Emulate Hardware

- When a VM tries to use its "hardware" (like read from a disk or send data), the hypervisor translates those actions into real hardware actions behind the scenes.

Step 6: Keep Everything Isolated and Secure

- Each VM runs in its own space, so if one crashes or gets infected, it won’t affect the others.
- VMs can’t access each other's data, keeping everything safe and separate.

REFERENCE

OpenVMS. In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/OpenVMS#cite_note-12

Stromasys. (n.d.). *OpenVMS Resources*. Retrieved from:
https://www.stromasys.com/resources/openvms/

VMS Software, Inc. (VSI). (n.d.). *Community License*. Retrieved from
https://vmssoftware.com/community/community-license/