Bahir Dar University

Bahir Dar Institute Of Technology Faculty Of Computing

Department : Software Engineering

Name : Abaynew Dessie

Id : 1600984 ***Section: A***

Submitted to : ***lec.Wendmu Baye***

TABLE OF CONTENTS

Installation of Haiku OS in Virtual Environment Tools

---

a. Introduction

Background and Motivation

Haiku OS is a modern, open-source operating system that traces its roots back to the **Be Operating System (BeOS)**, which was originally developed in the 1990s for multimedia applications. While BeOS was discontinued, it inspired the creation of Haiku, a project that aims to continue its legacy by providing a fast, responsive, and elegant system for personal computing.

Haiku is **specifically designed for simplicity, speed, and efficiency**, making it suitable for users who want a distraction-free environment or developers interested in system-level programming. Unlike mainstream operating systems like **Windows**, **macOS**, or even **Linux**, Haiku offers a unique take on **file systems, GUI design**, and **application interaction**, while still being POSIX-compliant and offering a familiar command-line interface.

Its core is built from scratch, using **C++**, and it features the **Be File System (BFS)**, which supports metadata indexing, journaling, and real-time file operations — features that are especially powerful for development and data management tasks.

Why This Project?

This project is not just an installation task — it's an opportunity to **gain real-world experience with operating systems**, virtualization, and system configuration. Here's why it's important:

---

1. Understand How to Install and Configure a Lesser-Known Operating System

Most students and professionals are familiar with installing Windows or Linux, but Haiku OS introduces new challenges:

- It uses a different file system (BFS), which is rarely encountered in traditional OS education.
- Some hardware or virtualization features may not be fully supported, requiring troubleshooting skills.
- The installation process offers insight into **bootloaders**, **disk partitioning**, and **system-level setup**.

By working with Haiku OS, students learn to **think critically and adapt** — skills that are essential in professional environments where they may encounter unfamiliar systems or legacy software.

2. Explore the Features, Performance, and Structure of Haiku OS

This project encourages you to look deeper into how Haiku works:

- Analyze its **boot time**, **memory usage**, and **interface responsiveness**.
- Study how it handles **process management**, **file indexing**, and **system responsiveness**.
- Understand the **user interface philosophy** of Haiku, which emphasizes minimalism, speed, and clarity.

This exploration develops your ability to **evaluate and compare operating systems** — a critical skill in fields like cybersecurity, software engineering, and systems administration.

---

3. Learn About Virtualization Tools and Their Usage

Virtualization is a key technology in modern computing:

- It allows you to **run multiple operating systems** on a single machine without affecting your host system.
- You gain experience with tools like **VMware** or **VirtualBox**, which are widely used in industry and education.
- Through this, you learn about **resource allocation**, **disk provisioning**, and **virtual hardware configuration**.

Understanding virtualization helps you prepare for careers in:

- Cloud computing (e.g., AWS, Azure, GCP)
- System testing and deployment
- Software development and QA

---

b. Project Objectives

1.Installing Haiku OS Successfully with a Virtual Machine

The main aim of this project is to get Haiku OS installed and up and running on my computer via a virtual machine. I'm using software like VMware or VirtualBox to create an environment in which I can test and research Haiku without risking my main system.

This part involves:

Downloading the official Haiku ISO image.

Setting up the virtual machine with the correct hardware settings (e.g., RAM, CPU, and disk).

Making the best decisions during setup so Haiku boots properly.

At the completion of this step, I will have a functional Haiku OS installed in a virtual machine to learn and play with.

2.Get to Know the Haiku File System (BFS)

Once Haiku is installed, I want to learn about how it saves files using something called the Be File System (BFS). It's not similar to file systems I've heard of (e.g., NTFS in Windows or ext4 in Linux) and is supposed to be fast and dynamic.

I will probe into issues such as:

- ✓ How directories and files are stored on disk.
- ✓ How metadata and file attributes are utilized.
- ✓ The handy tool that lets you search files in real time by their attributes.

- ❖ The objective is to become familiar with navigating and using BFS, and understanding why it is special.

3.Learn from Installing Process and Troubleshoot Problems

Things don't always go so smoothly when you install a new OS, especially one like Haiku that is not as well-known. So another gigantic part of this project is to watch what problems I run into, how I get around them, and document as I do it.

These could be things like:

- ✓ The OS not booting properly.
- ✓ Virtual hardware not being detected.
- ✓ Internet or screen resolution issues.

c. Requirements

i. Hardware Requirements:

- **CPU**: Intel or AMD processor with virtualization support (VT-x or AMD-V).
- **RAM**: Minimum 2 GB (4 GB or more recommended).
- **Disk Space**: At least 20 GB free for the virtual machine.
- **Display**: SVGA or compatible display adapter.

ii. Software Requirements:

1. Virtualization Software

In order to install and use Haiku OS on your computer without replacing your current operating system, you'll need virtualization software. This type of software enables you to set up a "virtual computer" inside your real one. Two of the most popular and free ones are:

- **VMware Workstation Player**

This is a stable and user-friendly virtualization software for Windows and Linux.

✅ How to Download and Install VMware Workstation Player (Example: Version 17.6.3)

- **Go to the Official Website**

- **Get your browser out and head on over to VMware's official site:**

https://www.vmware.com

- **Get VMware Workstation Player**

- ✓ **Go to the Products page, click on VMware Workstation Player. Make sure not to choose the "Pro" version unless you need the added features.**
- ✓ **Pick Your OS**
- ✓ **Choose the one based on your operating system (most likely Linux or Windows).**
- ✓ **Click "Download Now"**
- ✓ **Click the download button to get the installer file. It may be named something like VMware-player-17.6.3.exe for Windows.**

Run the Installer

Once it's downloaded, double-click the file to start the installation. Follow the steps shown on screen — they're usually straightforward:

- ✓ **Accept the license agreement**
- ✓ **Choose the installation folder**
- ✓ **Enable/disable automatic updates**
- ✓ **Click "Finish" when done**

- **After installation, you'll be ready to create your first virtual machine and install Haiku.**

2. Haiku OS ISO File

The second thing you need is the operating system itself: Haiku OS. Since you'll be executing it within a virtual machine, you need the ISO image file, which is an electronic representation of a bootable disc.

✅ How to Download the Haiku OS ISO (Latest 64-bit Version)

- **Go to the Official Download Page**

- ✓ **With your browser, go to:**

  https://www.haiku-os.org/get-haiku/

- **Select the Stable Release**

- ✓ **Look for the 64-bit under the newest stable release (for instance, R1/beta5 is the newest version at the time of writing).**

- **Select the "Anyboot ISO" Format**

- ✓ **This format is ideal for both USB and virtual machines.**
- ✓ **Click on the "Anyboot ISO" download link.**

- **Select a Mirror Nearby**

- ✓ **When the site prompts you to choose a mirror, select one geographically close to where you are. This makes the download quicker and more stable.**

- **(Optional but Highly Recommended) Verify the Downloaded ISO File**

- ✓ **You can check the SHA256 checksum to confirm your file is not damaged or tampered with.**
- ✓ **The website provides you with a checksum string — run something like sha256sum (Linux/macOS) or PowerShell (on Windows) to confirm.**

- **After you download this ISO file, here's what you'll "insert" into your virtual machine and begin installing Haiku OS.**

d. Installation Process

This tutorial provides a step-by-step, detailed installation guide for Haiku OS on a virtual machine (VirtualBox or VMware). Use this to avoid common issues.

Step 1: Download Haiku OS

- **Get the correct ISO:**

- **Visit Haiku OS Official Downloads.**
- **Choose:**

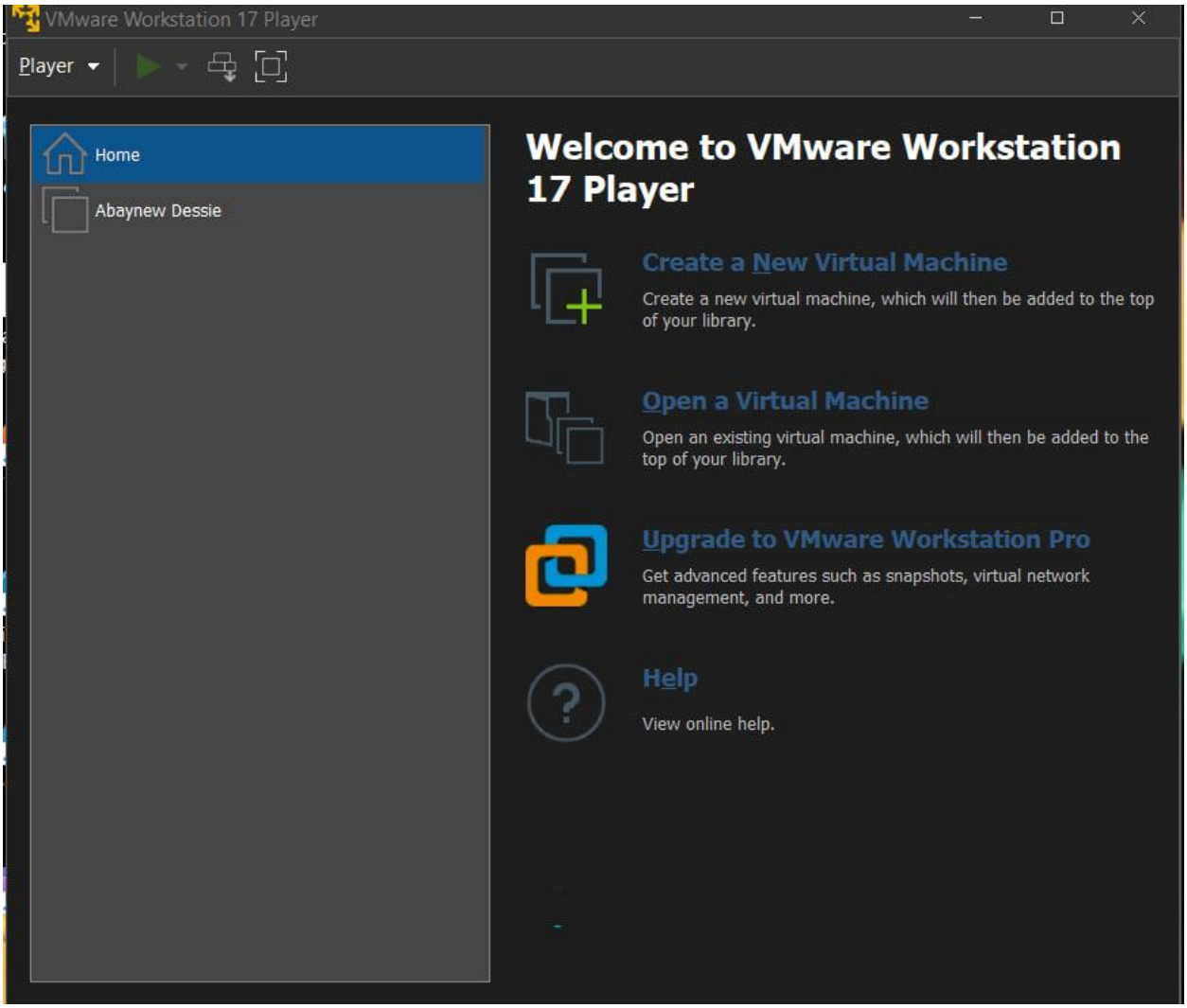
- ✓ **Stable Release**
- ✓ **Nightly Build**

- **Save the .iso file to somewhere that you can easily access (e.g., Downloads/Haiku_OS).**

Step 2: Create a New Virtual Machine

❖ In VMware Workstation:

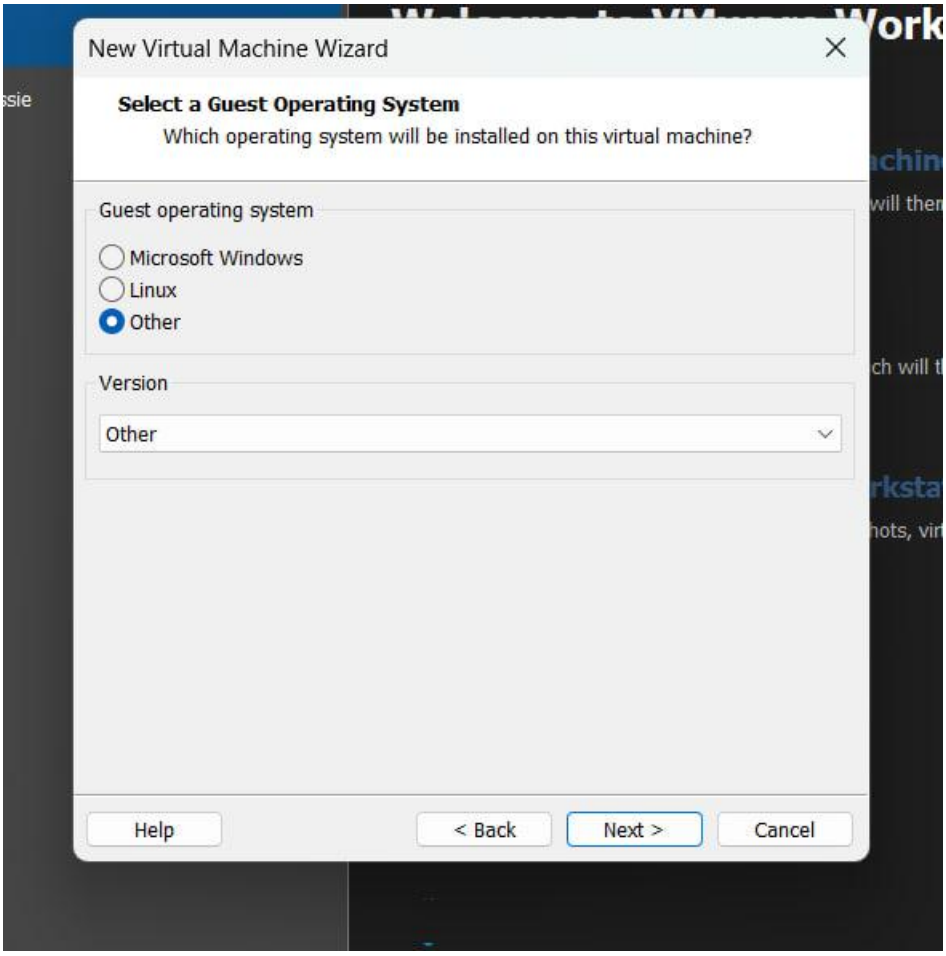
- **Open VMware Workstation Pro.**
- **Click on "Create a New Virtual Machine".**



✓ **Select:Custom (advanced) (Typical mode might not be appropriate for Haiku).**

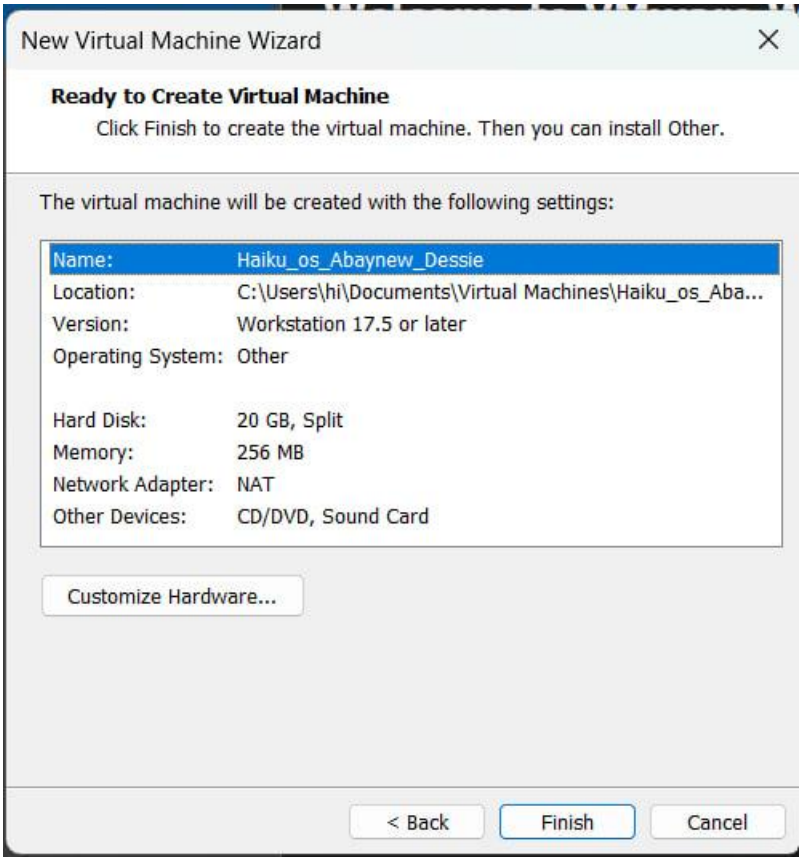
➢ **Set settings:**

- **Guest OS: Choose "Other" → "Other 64-bit" (Haiku isn't in the default list).**
- **VM Name: Haiku_OS_Abaynew_Dessie**



- **Memory: Assign 2GB (2048 MB) minimum (4GB if you have free RAM).**
- **Network: NAT (default, fine for internet access).**
- **Disk:**

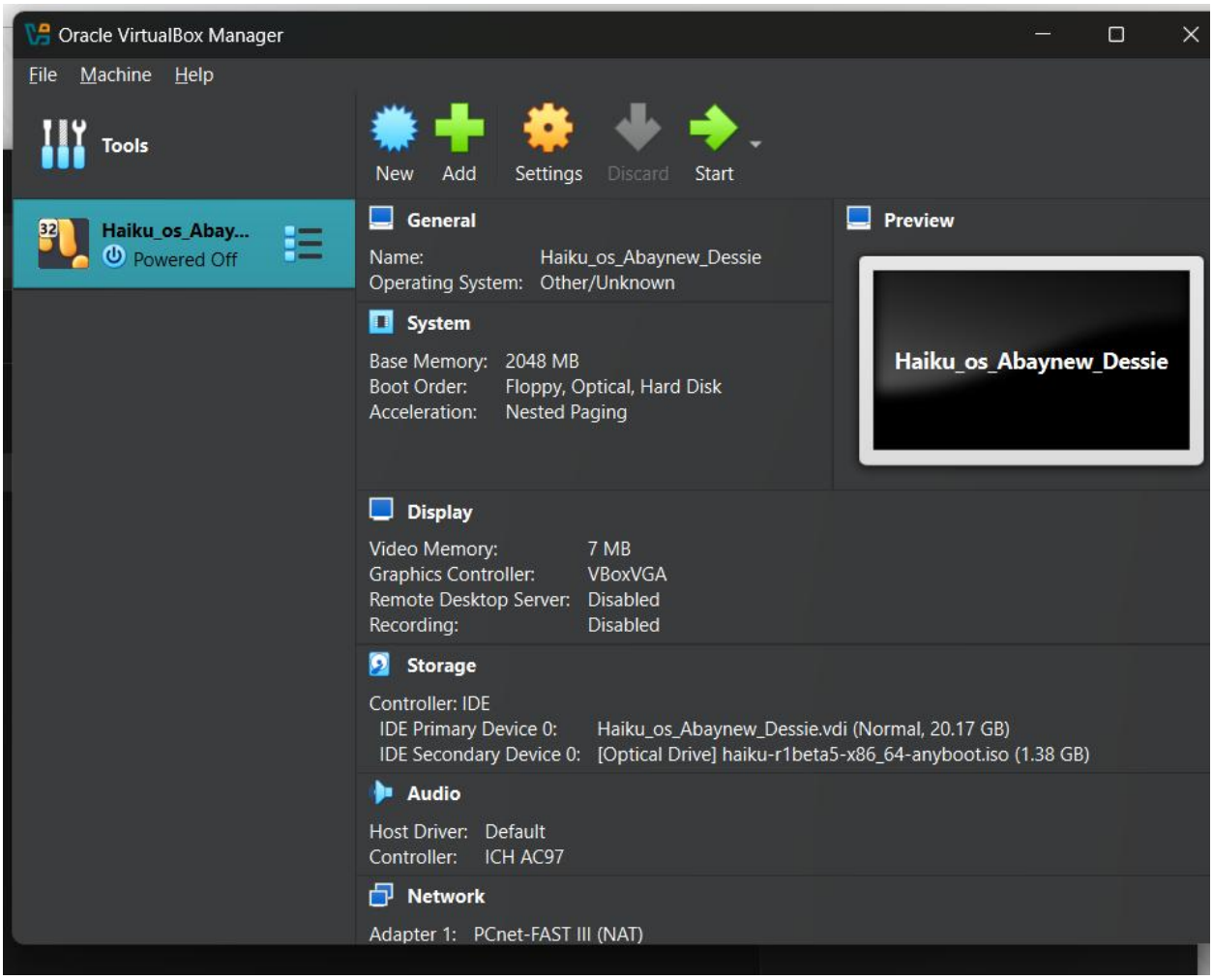
✓ **20GB virtual disk (select "Store as a single file" for better performance).**
✓ **Disk Type: IDE (Haiku may not properly recognize SCSI/SATA).**



❖ In Oracle VirtualBox:

- **Open VirtualBox → Click "New".**
- **Configure:**

- ✓ **Name: Haiku_OS_Abaynew_Dessie**
- ✓ **Type: Other**
- ✓ **Version: Other/Unknown (64-bit)**
- ✓ **RAM: 2GB+**
- ✓ **Hard Disk: Create a virtual hard disk now → VDI (dynamically allocated, 20GB).**



Step 3: Attach the Haiku ISO File

VMware:

- ✧ **Select your VM → "Edit virtual machine settings".**
- ✧ **Go to CD/DVD (SATA) → Mark "Use ISO image file" → Browse for your Haiku.iso.**
- ✧ **Mark "Connect at power on".**

VirtualBox:

- ✧ **VM → Settings → Storage.**
- ✧ **Under Controller: IDE, click on the empty disk icon → Choose a disk file → Haiku ISO.**

Note: Plug in the ISO before booting the VM!

Step 4: Install Haiku OS

✧ **Boot the VM → It will boot from the ISO.**
✧ **In Haiku Boot Menu, select:**

✓ **"Boot Haiku" OR**
✓ **"Install Haiku"**

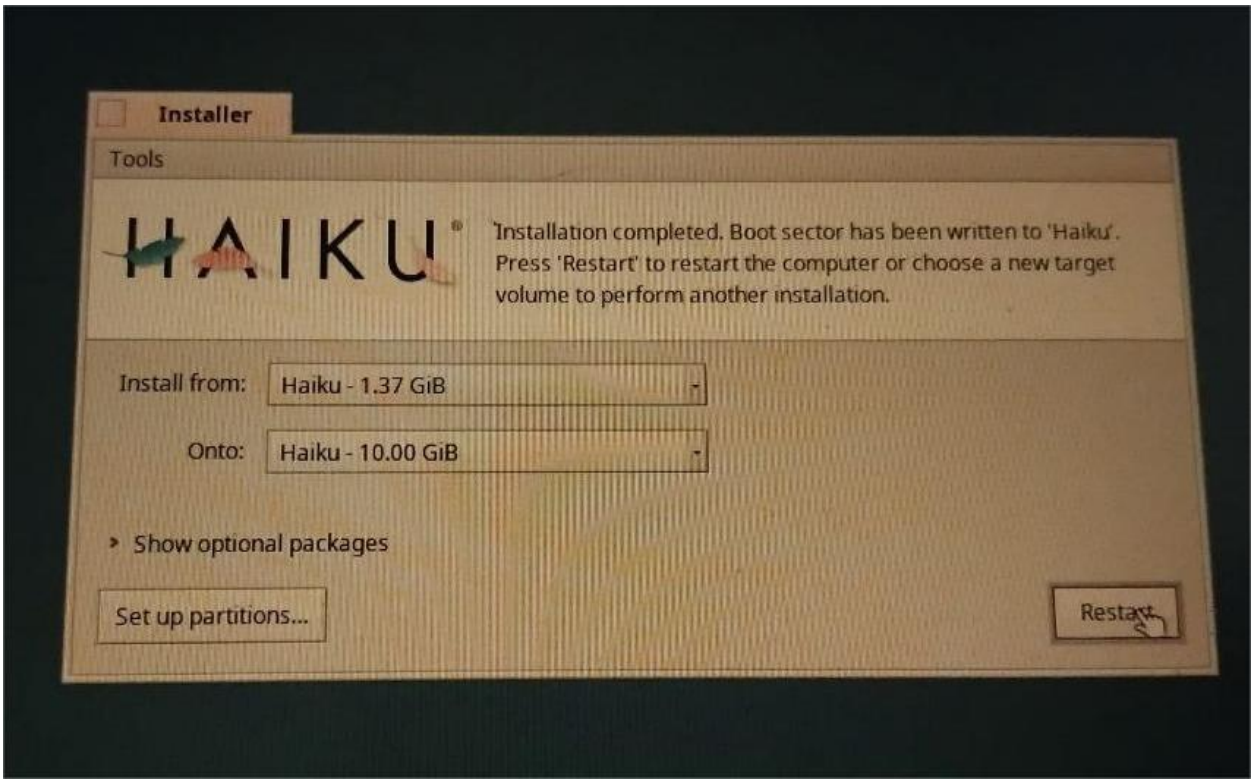
✧ **Execute Installer:**

✓ **Open "InstallHaiku" on the desktop.**
✓ **Select your virtual disk (e.g., 20GB VBOX HARDDISK).**
✓ **Select "Full Installation" (recommended).**

✧ **Filesystem Setup:**

Format with BFS (Be File System) (native filesystem of Haiku).

Confirm and proceed.



After set up partitions like this .

Restart it.

Wait for installation to complete—takes ~5-10 minutes.)

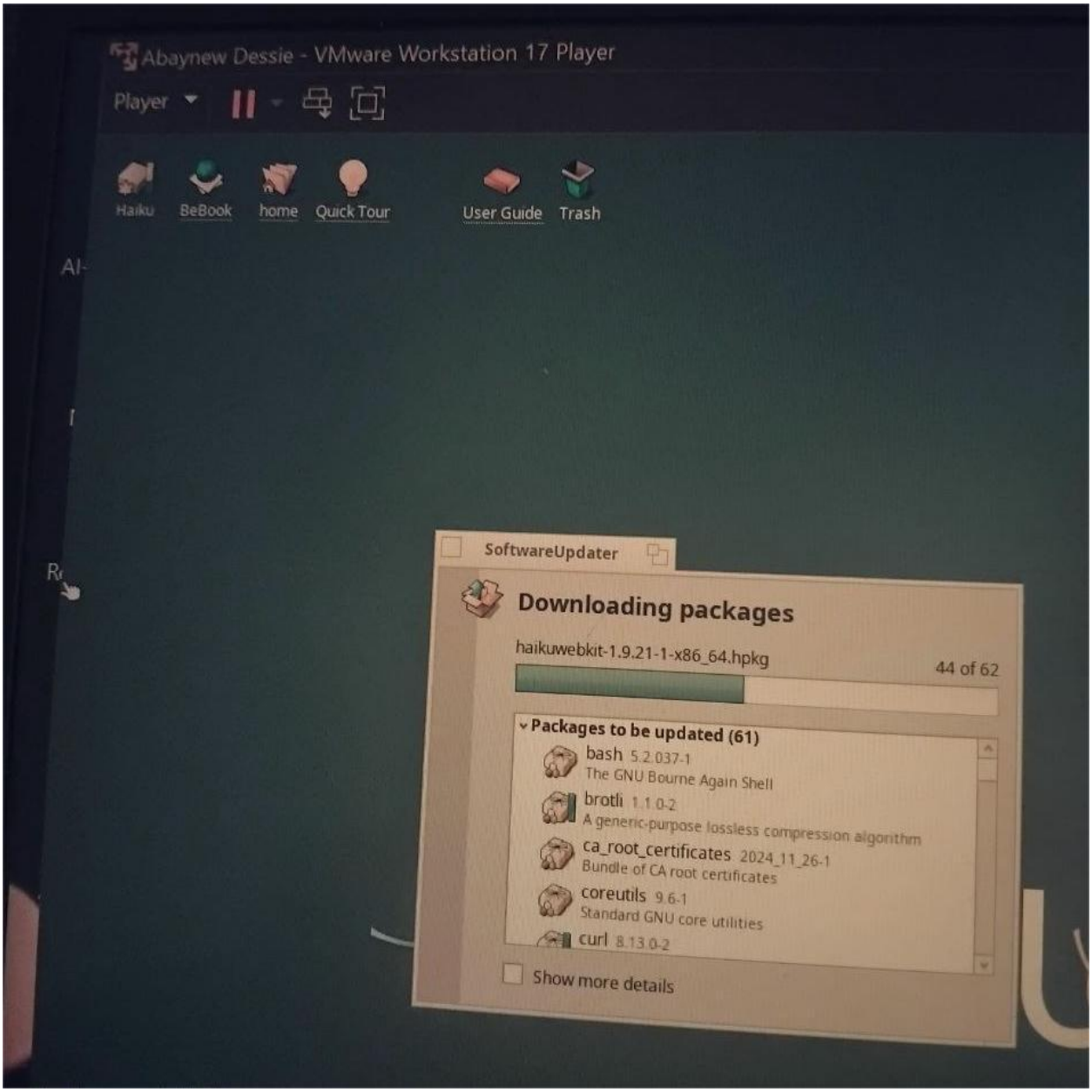
Step 5: Set Up User Account & Final Setup

Set Up User Account:

Username: Abaynew dessie

Real Name: Abaynew Dessie

Password: (optional, but recommended for security).



After Full Installation:

Press "Reboot" when prompted.

Unmount the ISO from VM settings so it won't automatically re-run the installer.

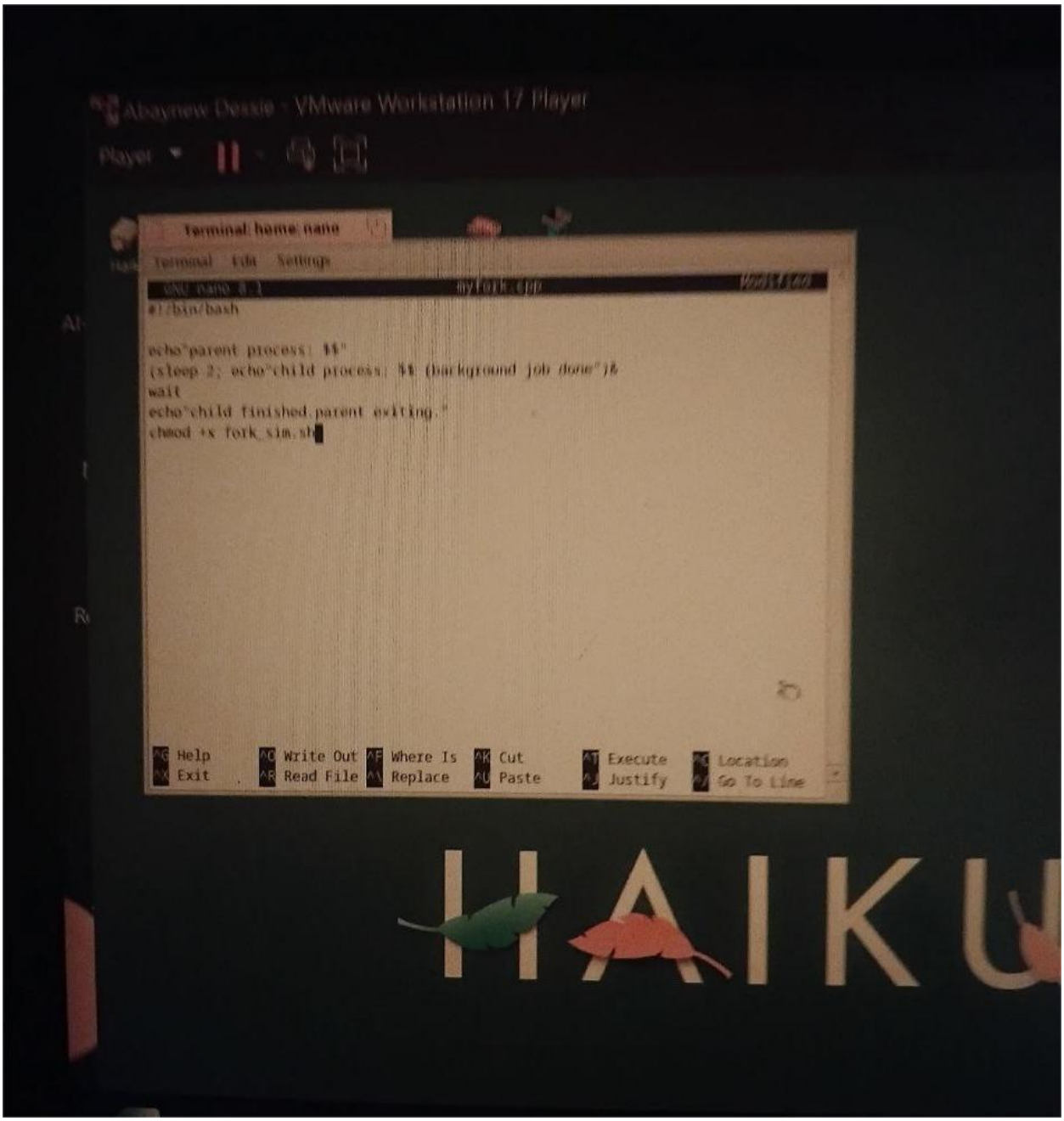
Tip: If the VM re-boots into the installer, shut it down, unmount the ISO, and re-start.

Step 6: Initial Boot & Confirmation

After re-boot, you should now obtain the Haiku desktop.

Confirm:

Open Terminal → Run uname -a → Should give Haiku.



- **If Haiku doesn't boot, check BIOS as the option selected in VM settings.**

E. Common Issues Encountered

During the installation of Haiku OS inside a virtual machine environment, some technical glitches are possible that can prolong the installation. A commonly faced issue is when the virtual machine fails to boot the ISO file, usually displaying a black screen, "no bootable device" alerts, or endless reboot loops. This problem typically occurs as a result of inappropriately set boot priorities under the VM BIOS settings, such that the CD/DVD drive has not been selected as the first boot device. In addition, the ISO file itself might be corrupted during the download or is not properly mounted to the virtual drive, and compatibility issues between the virtualization software settings (e.g., UEFI vs Legacy BIOS modes or Secure Boot) and Haiku's requirements can complicate matters even further.

The second most frequent problem occurs when the virtual disk is not recognized by the installer, where it shows messages that it found no disks or reports unallocated space. This is commonly caused by storage controller incompatibilities, particularly when the VM defaults to SCSI/SATA controllers that Haiku may not support. Disk format issues, including prior partition table conflicts or uninitialized disk space, and improper virtual disk configuration settings (e.g., disk size is too small or erroneous allocation methods) can also prevent disk detection from being successful.

The actual installation can hang or crash at various stages, typically stalling during file copy or experiencing unexpected VM rebooting. These intermit more often than not are directly related to bad resource planning, where inefficient RAM (lesser than 2GB) or CPU core assignments cause performance bottlenecks. Software conflicts within the hypervisor environment, particularly involving the graphics acceleration feature or memory management drivers, as well as potential corruption of the installation media itself, are additional points of failure that can cause intermitting of the installation process.

Post-installation network problems will most likely render the user internet-inaccessible, reflected in missing network icons or non-functional interface setups. The root causes will generally be due to virtual network adapter type vs. Haiku driver support mismatches, misselected NAT/Bridged mode settings, or the absence of important virtualization utilities like VMware Tools or VirtualBox Guest Additions. Input device failures, in which keyboards or mice do not respond within the VM, typically result from controller emulation conflicts between PS/2 and USB devices, lack of HID drivers, or focus management issues between host and guest operating systems.

Boot failure following a successful install is yet another category of problems, ranging from missing kernel errors to boot loops that never end. These are usually signs of bootloader installation problems, corruption of the filesystem in the BFS implementation, or virtual hardware abstraction failure. Graphical glitches or uncomfortably low resolutions plague users on a regular basis, mainly due to limitations in default VESA drivers, allocations of VRAM that are too small, or conflict between Haiku's native display system and the virtualization platform's graphics emulation.

Sound functionality will completely fail or provide distorted outputs, usually due to unsupported emulated audio hardware (AC'97 vs HDA) or missing driver pieces within Haiku's sound system stack. Failures in mounting the filesystems blocking proper access to disks are usually caused by BFS journal corruption, partition misalignment, or inconsistencies in virtual disk images. Lastly, overall performance and stability problems could arise from resource limitations, software incompatibility, or virtualization layer overheads inherent to the virtualized environment, taking the forms of system slowdown, application failures, or memory mismanagement.

F. Solution.

For boot failures from ISO, check first the BIOS settings of the VM for proper boot order with CD/DVD highest in priority. The ISO itself may be validated using checksum validation tools to verify that it completely downloaded free of corruption. Within virtualization packages like VMware or VirtualBox, manually select within the settings "Legacy BIOS" rather than UEFI because Haiku operates more smoothly under classical BIOS emulation. For VirtualBox users, boot issues are typically resolved by mounting the ISO through the IDE controller rather than SATA, while VMware users should make sure the "Connect at power on" box is checked for the virtual CD/DVD drive. If Secure Boot is enabled in the host system's virtualization settings, disabling this feature tends to allow Haiku to boot without issue.

For installer disk detection issues, the primary solution is to change the virtual disk controller type to IDE in the VM settings, as Haiku supports this older standard natively more easily. Pre-install, run Haiku's built-in DriveSetup utility to completely wipe and reformat the virtual disk with the Be File System (BFS). In VirtualBox, the virtual disk creation with "pre-allocate full size" rather than dynamic allocation will prevent detection issues, and those who use VMware need to ensure that their virtual disks are created in compatibility mode with earlier VMware versions. If the disk remains undetected, the VM recreation with these modified settings tends to rectify the problem.

Freezes and crashes on installation are typically due to resource allocation issues, so increasing memory on the VM to at least 2GB (4GB if possible) and assigning two CPU cores provides more stable performance. Disabling unnecessary features like 3D acceleration and USB controllers at install time reduces potential points of failure. Operating the stable version of Haiku rather than nightly builds minimizes instability, and having the host system possess adequate resources available (not overcommitted to alternative VMs or applications) prevents resource contention. In the event of persistent freezing, trying an alternate virtualization platform (switching from VMware to VirtualBox or vice versa) can isolate software-specific issues.

Post-installation network connectivity issues involve verifying the network adapter settings of the VM - the "PCnet-FAST III" adapter for VirtualBox or "E1000" for VMware tends to perform optimally with Haiku. It is important to verify that the network mode is configured for NAT (for basic internet connectivity) or Bridged (to be visible on the local network) as required. Installing the appropriate guest additions package for the virtualization platform provides better drivers and functionality, though with the caveat of having to download Haiku-specific versions from community repositories where available. Early network troubleshooting in Haiku involves verifying the interface presence in ifconfig output and testing with ping to known-good IP addresses before verifying DNS resolution.

Input device issues generally get resolved by changing the pointing device type under VM settings to "USB Tablet" in VirtualBox or deactivating "absolute pointing device" in VMware. The host key (typically Right Ctrl) releases captured keyboard and mouse input to the host operating system when needed. Installation of guest additions improves integration of input devices, while looking for conflicting USB controller settings in the VM configuration prevents enumeration problems. In the case of keyboard mapping issues, manually selecting the proper key layout during Haiku's first setup ensures proper character input.

Installation boot failures typically require bootloader reconfiguration by restarting with the installation media one more time and using the bootman tool to repair the installation. Verifying the disk partition is marked as active and the boot flag is properly set fixes most booting issues. In VirtualBox, enabling EFI mode (despite not being Haiku's default) can assist with problematic boots persisting, and placing the virtual disk at the top of the boot order keeps the system from attempting to find elsewhere for bootable media. When experiencing kernel panics, reinstalling with verbose logging on will help pinpoint the driver or service responsible for the crash.

Display resolution limitations are improved by installing the appropriate guest additions package with video drivers optimized for the virtual environment. Manual setup of the Haiku framebuffer mode through the /boot/system/settings/kernel/drivers.conf file allows custom resolutions in case of detection failure. More video memory allocated to the VM, at least 64MB, provides more resources to the display system, while disabling unnecessary 3D acceleration features reduces conflicts for Haiku.

Audio support includes selecting the proper emulated sound card in VM settings - in most cases, Intel HD Audio is the preferred option. Verifying that an audio server is running in Haiku with the Media preferences panel and examining mixer settings ensures accurate output routing. In case of stutters or distortion in audio, reducing the sample rate in Haiku's sound preferences to 44100Hz stabilizes it, as does disabling the audio input devices that are not in use.

Filesystem mounting errors typically entail running fsck_bfs on the affected volume from a live CD session to check and repair filesystem damage. Correct disk partition alignment (starts at sector 2048 for modern systems) prevents disk performance issues in addition to mounting errors. For virtualized environments, using fixed-size rather than dynamically allocated disks reduces the risk of filesystem corruption, while regular disk image maintenance (defragmentation where the host operating system allows it) maintains optimum performance.

General performance optimization involves allocating enough CPU cores (without host resource overcommit) and enabling nested paging or VT-x/AMD-v acceleration in the host BIOS. Overhead is minimized by disabling unneeded virtual hardware devices like unused COM ports or floppy disk drives, while configuring the VM's storage controller for write-back caching (with proper host backups) improves disk performance. Monitoring system resources through Haiku's ProcessController makes it easy to identify specific applications or services causing performance bottlenecks, allowing targeted troubleshooting. Regular updates of the virtualization platform and Haiku OS allow access to new performance improvements and bug fixes.

G. Be File System (BFS)

Be File System (BFS) is a high-performance file system that was initially developed for BeOS and is currently the native file system of Haiku OS. BFS was architected for multimedia programs and real-time operation and has a number of advantages over traditional filesystems. Its advanced indexing capabilities enable nearly instantaneous file

searching even when one needs to work with directories containing millions of files - a far superior performance than FAT32's limitations or even ext4's capabilities when dealing with extremely large directories. The filesystem has built-in journaling, similar to NTFS and ext4, which ensures data consistency in case of abrupt power shutdown or system crashes and also enables minimized recovery times.

One of the strongest features of BFS is that it can handle custom file attributes, which allow users to attach metadata like author names, project tags, or media specifications to files. This database-like file handling makes it particularly valuable for media professionals and developers who handle large sets of files. The filesystem is also efficient for small files using its dynamic inodes to avoid other systems' space wastage. BFS does great with its intended use cases, though it suffers from a few caveats of its own, including no native support outside of Haiku/BeOS environments and no features like snapshoting or SSD optimization software. BFS remains ideally suited to Haiku's architecture to provide the low-latency performance and robust metadata support that the operating system requires. Its construction is a mindful balance between general-purpose computing demands and particular multimedia workflows and so is a fundamental component of Haiku's distinguishing capabilities. Continuing filesystem development can eventually remove present limitations while still maintaining its underlying strengths of speed, reliability, and flexibility.

H. Advantages and Disadvantages

Haiku OS provides an interesting but specialized operating system option with interesting strengths and limitations. Its best feature is its minimalist approach, providing exceptional responsiveness even on older hardware, with system requirements far lower than modern operating systems. This is because it has a lean design, which provides immediate boot times and smooth operation on computers with as little as 512MB of RAM. The system's multimedia strengths, inherited from its BeOS roots, provide better low-latency performance for video playback and audio processing, making it of particular interest to creative professionals. As an open-source project, Haiku benefits from community-developed clean code and user control, a welcome respite from the commercial operating systems' increasing bloat and mandatory software updates. The interface is uncommonly consistent and uncluttered, shunning the fragmentation found in Linux distributions but delivering a more foreseeable user experience than Windows.

Yet, Haiku is beset by serious problems that restrict its use in mainstream markets. Hardware compatibility continues to be an issue, with intermittent driver support for new components such as WiFi cards and GPUs, requiring workarounds for minimum functionality. The software ecosystem, while growing, lacks significant applications, so

users turn to alternatives or complicated compatibility hacks. Business, development, and creative work will experience considerable gaps in tools versus mature platforms, which professional users will feel. Unlike commercial operating systems or enterprise Linux distributions, Haiku does not include long-term support commitments, and users are responsible for system updates and security patches. These limitations now position Haiku best suited for hobbyists, retro computing projects, and niche applications where its efficiency and simplicity benefits outweigh its lack of features. Not yet ready to replace mainstream operating systems for general use by the majority of people, Haiku continues as an interesting alternative in the open-source world, particularly for those who value performance and aesthetic appeal over broad compatibility.

I. Conclusion

Installation and use of Haiku OS in a virtualized environment is an exercise that provides much more than naked technical tinkering - it provides insight into a competing operating system design philosophy that prizes efficiency, elegance, and user-centered functionality first and foremost. Virtualizing Haiku gives end-users firsthand experience with its amazingly lightweight design, seeing how an OS can boot in sub-10-second times and provide smooth performance even on humble virtualized hardware configurations. This exercise also springs to life on going through Haiku's native Be File System (BFS) to understand just how meticulous design of a filesystem can have profound implications on system responsiveness by allowing features like instant metadata lookups, crash-safe journaling, and file attributes extensibility.

Virtualization experience also demonstrates Haiku's persistence of relevance due to its BeOS-inspired design even in contemporary computing environments. Although the OS shows some weakness in driver support and software choice, its solidly integrated design - avoiding the creeping complexity of general-purpose operating systems - is a priceless lesson in system integrity and performance. For computer science students and programmers, messing around with Haiku in a virtual machine is hands-on learning in filesystem optimization, real-time system design, and feature versus performance design tradeoffs. Moreover, it violates conventional expectations of what an operating system should be, and provokes new system design.

Apart from its technical merit, virtualizing Haiku preserves and extends the legacy of a major alternative model of computing. It allows new generations of users to enjoy and learn from a system that once pioneered concepts now taken for granted in modern OSes, yet retain it accessible through safe, reversible virtualization. Whether for educational, research, or curiosity's sake, running Haiku within an emulated setting is a tribute to computer history and an educative study of operating system principles that remain valid today. The experience comes in the end to demonstrate the manner in which such alternative operating systems as Haiku continue to augment the innovation and diversity of the computing space despite an era dominated by a few powerful platforms.

L. Recommendations and Future Work

One particularly rewarding area is to delve deeper into the original development environment and application programming interfaces (APIs) of Haiku. The platform offers a thoroughly documented set of tools like the Haiku API that was backward compatible with BeOS but with enhanced modern features. Future developers may experiment with creating apps that leverage Haiku's multimedia processing and real-time capabilities, maybe porting well-known open-source packages or developing new utilities tailored to the platform's capabilities. The development kit, although different from mainstream fare, is an excellent means of studying alternative approaches for graphical user interfaces and system services. Network settings and software installation are another important area to explore further.

Users can explore Haiku's package management system, which is less extensive than Linux distributions but includes the essentials and a few applications. Experimenting with various networks - from basic DHCP configurations to more advanced static IP configurations - should provide some insight into the networking stack and how Haiku supports modern protocols. Probing Haiku's hybrid kernel design on networked systems could be fascinating, particularly for latency-sensitive applications. Additionally, attempting to create and implement open-source software that is coded for other operating systems can give useful information on Haiku's POSIX compatibility and what the system could become with additional development. For those who want to make a more impactful contribution, joining the Haiku open-source community is the best future endeavor.

The project welcomes contributions of all types, ranging from documentation and improvement to kernel development, with a particular requirement for driver development to enhance hardware compatibility. New developers can start by

testing nightly builds, reporting bugs, or helping port core applications. More experienced developers can tackle tasks like improving virtual machine integration, filesystem performance, or creating new system utilities. The community includes strict documentation for would-be contributors, like coding standards and style guides, that will make the developers with whatever level of expertise can contribute substantively to the ongoing development of Haiku. New chat

---

2: Virtualization in Modern Operating Systems Like Haiku os

1. What is Virtualization?

Virtualization is a fundamental computing technology that makes it possible to run multiple operating systems at once on a single physical computer through the creation of isolated virtual environments. In effect, this technology works by representing hardware resources through a dedicated layer of software referred to as a hypervisor, which allocates processor capacity, memory, disk space, and other hardware resources to each VM according to need. In our project, we're utilizing virtualization to install and experiment with Haiku OS - an open-source, lightweight operating system derived from BeOS - and have our primary operating system (e.g., Windows, macOS, or Linux) completely unaffected and intact. It has several significant advantages: it eliminates the risk of data loss or system instability that can come with an uncomplicated hardware installation, enables easy experimentation among different configurations with snapshots and clones, and enables side-by-side comparisons of operating systems with no need for discrete physical machines.

The virtualization process creates a standalone virtual computer that behaves exactly like physical hardware, such as virtual processors, RAM, storage drives, and network interfaces. When we run Haiku OS in this arrangement, it has the illusion that it is directly connected to computer hardware, when in reality, the hypervisor diligently handles all resource allocation and prevents host and guest systems from interfering. Such isolation is particularly beneficial when experimenting with alternative operating systems like Haiku since it allows us to

safely try out its new features - e.g., its responsive interface, fast Be File System (BFS), and multimedia functions - without changing our main working environment permanently. Modern virtualization software like VirtualBox and VMware also have easy-to-use interfaces with which to manage and create these virtual machines, enabling the technology to be accessed by even non-technical users but maintaining the heavy-hitting features for experienced users.

---

2. Why Virtualize Haiku OS?

a. Safe Testing Environment

Virtualization offers a totally isolated testing environment for Haiku OS without endangering your core operating system. As Haiku is under active development, certain hardware drivers and pieces of software functionality can be unstable or experimental in nature. Running it directly on hardware could result in loss of data, bootloader conflicts, or system crash—particularly if there are partition errors. With a virtual machine (VM), all changes are contained in a self-contained sandbox, so any software bugs, misconfigurations, or failed updates in Haiku won't touch your host computer (Windows, Linux, or macOS). This makes virtualization ideal for students, developers, and enthusiasts who want to experiment with Haiku's unique features—such as its Be File System (BFS) and real-time capabilities—without risking their main computer.

b. No Dual Booting Required

The traditional methods of testing a new OS typically involve dual-booting, which requires partitioning the hard drive and installing the second operating system alongside the original one. This is a complex and risky procedure that can lead to bootloader corruption, accidental data loss, or system instability. Virtualization eliminates all these problems because Haiku OS can be installed as an application within your current OS. With VirtualBox or VMware software, you can have Haiku running in a window just like any other program—without rebooting or modifying your disk partitions. This is much more convenient for occasional users and

testers who require quick access to Haiku without the hassle of handling multiple boot entries or resizing storage partitions.

c. Easy to Revert and Reset

One of the biggest advantages of virtualization is having the ability to reverse mistakes immediately. Virtual machines allow for snapshots, which freeze the OS at a given point exactly as it is. If Haiku crashes, gets misconfigured, or becomes useless due to experimental changes, you can restore an earlier snapshot in seconds—essentially "reversing time" back to when the system worked. It is a lifesaver for learning and debugging, as it allows students and developers to:
Test risky modifications (e.g., kernel hacks, driver installations) without permanent effect.
Experiment with different settings (e.g., filesystem layouts, network configurations) and easily revert if needed.
Avoid lengthy reinstallations—instead of rebuilding from scratch, just roll back to a fresh snapshot.
While physical installs, with mistakes potentially requiring full OS reinstalls, might be infuriatingly time-consuming and error-ridden, virtualization gives a seamless and risk-free experience for experimenting with Haiku OS.

Summary of Benefits

- Safe Testing

Unrelated to host OS
No risk of destroying your main system

- No Dual Boot

Acts like an application
No rebooting or partitioning required
Snapshots

- Immediate undo when making mistakes

Best for experimenting and learning

---

3. Virtualization Tools Used for Haiku OS

To run Haiku OS in a virtual environment, we use a **Type 2 hypervisor**, which runs on top of a host OS:

Tool	Description	Role
VMware Workstation Player	A free virtualization software for Windows/Linux	Creates the virtual environment for Haiku
Oracle VirtualBox	Open-source and cross-platform virtualization software	Also supports Haiku OS installation

Both tools allow you to:

- Create a **virtual machine**
- Allocate resources (RAM, CPU, Disk) to Haiku OS
- Mount the Haiku ISO image and **boot from it**

---

4. How Virtualization Works in the Case of Haiku OS

❖ Virtual Hardware Building

When setting up a virtual machine (VM) for Haiku OS, the hypervisor (such as VirtualBox or VMware) builds a virtual computer with all hardware components—save that they are all software-based. The virtual hardware includes:

✓ Virtual CPU & RAM: The hypervisor allocates a portion of your real CPU and memory to the VM. With Haiku, only 1-2 CPU cores and 2GB of RAM is required to run.

✓ Virtual Hard Drive: Instead of modifying your real disk, the VM loads and saves its virtual disk file (e.g.,.vdi or.vmdk), which appears as an independent storage drive.

✓ Virtual Display & Network Card: VM emulates normal graphics (VESA) and network (Intel PRO/1000 or AMD PCnet) adapters so Haiku can run without the need for special drivers.

❖ Booting from Haiku ISO

To install Haiku, the VM emulates the downloaded ISO file as if it were a physical CD/DVD

- ISO Mounting: The .iso file is mounted by the hypervisor to the VM's virtual optical drive, mimicking a bootable disc.

- Booting Process: On booting the VM, it loads Haiku ISO first (like a regular PC). Haiku boot menu is presented to users, where they are given the option to:

- ✓ Test Haiku live (not installed).

- ✓ Begin installation to virtual disk.

- Installation Launch: The installer runs inside the VM, with total isolation from the host OS.

This step causes Haiku's installation process to behave just as it would on actual hardware—though without any danger to your actual machine.

❖ Installing Haiku on a Virtual Disk

During the installation process, Haiku interacts with the virtual disk only:

- ✓ Disk Formatting: The installer identifies the VM's virtual drive and partitions it with BFS (Be File System)—Haiku's high-performance, optimized native filesystem for speed and metadata.

- ✓ OS Installation: Haiku system files are copied to the virtual disk and create a fully independent Haiku environment.

- ✓ Bootloader Setup: Haiku boot manager is placed only in the VM and has no interaction with the host system boot process.

❖ Running Haiku OS as a Guest OS

After installation, Haiku exists as a guest OS inside the VM:

- ✓ Seamless Interaction: Haiku's terminal, GUI, and applications may be used exactly as on a native PC—within a resizable window.

- ✓ Resource Management: CPU, RAM, and storage between Haiku and the host OS are dynamically managed by the hypervisor.

- ✓ Network Access: Haiku may connect to the internet via the VM's virtual NAT or bridged networking, yet be firewalled from the host.
- ✓ Snapshot Safety: If Haiku crashes or becomes misconfigured, you can recover a snapshot to undo instantly.

---

5. Advantages of Virtualizing Haiku OS

Advantage	Explanation
Safe Exploration	Students can explore system structure and behavior without damaging the host OS.
Faster Testing	Developers can test Haiku apps, drivers, or features without using a separate physical machine.
Isolation	If Haiku crashes or fails, it doesn't affect Windows/Linux.
Portability	The entire Haiku VM can be copied to other computers.
Snapshot Support	Save the current state of Haiku and revert back when needed.

---

6. Limitations While Using Haiku in Virtual Machines

Limitation	Explanation
Limited Hardware Acceleration	Some graphics and audio features may not work properly.
Networking Issues	Networking in Haiku inside a VM may need configuration (e.g., bridging,

Limitation	Explanation
	NAT).
Performance Overhead	Haiku may run slower inside a VM compared to a real installation.

---

7. Conclusion

Virtualization offers a simple and secure way of trying Haiku OS without necessarily having to buy new hardware or alter the underlying system configuration. With the use of virtual machines, one can download, execute, and test Haiku in a sandboxed environment, which prevents possible risks such as data loss or system crashes. This method is particularly helpful for learners of operating system concepts, researchers exploring non-traditional computing paradigms, and programmers developing Haiku-native applications. Since virtualization allows for quick setup, restore from snapshots, and host-guest switching without interruption, productivity is improved and potential interruptions minimized. Hence, virtualization is a safe and efficient means to tackle niche operating systems like Haiku, opening up to accessibility and experimenting in a pure environment.

Implementing System Calls in Haiku OS – `fork() system call`

Overview:

The `fork()` system call is used to create a **new process** by duplicating the calling process. This is a fundamental mechanism in UNIX-like operating systems, including Haiku OS, which follows POSIX standards.

When `fork()` is called:

- A new process (child) is created.
- The child receives a **copy** of the parent's memory, file descriptors, and execution context.
- The return value distinguishes the processes:
  - `0` for the **child**
  - The **PID of the child** for the parent
  - `-1` if the fork fails

Why It Matters:

- Essential for multitasking and multiprocessing.
- Used in combination with `exec()` and `wait()` to manage process trees.
- Demonstrates how operating systems manage resources across processes.

Example: `fork()` in Haiku OS using C/C++

```
#include <unistd.h>
#include <stdio.h>
#include <sys/types.h>

int main() {
    pid_t pid = fork();     if (pid < 0) {

        perror("fork failed");
        return 1;
    } else if (pid == 0) {

        printf("this is the child process! PID: %d\n", getpid());
    } else {
                printf("this is the parent process! Child PID: %d\n", pid);
    }

    return 0;
}
```

Compiling and Running in Haiku:

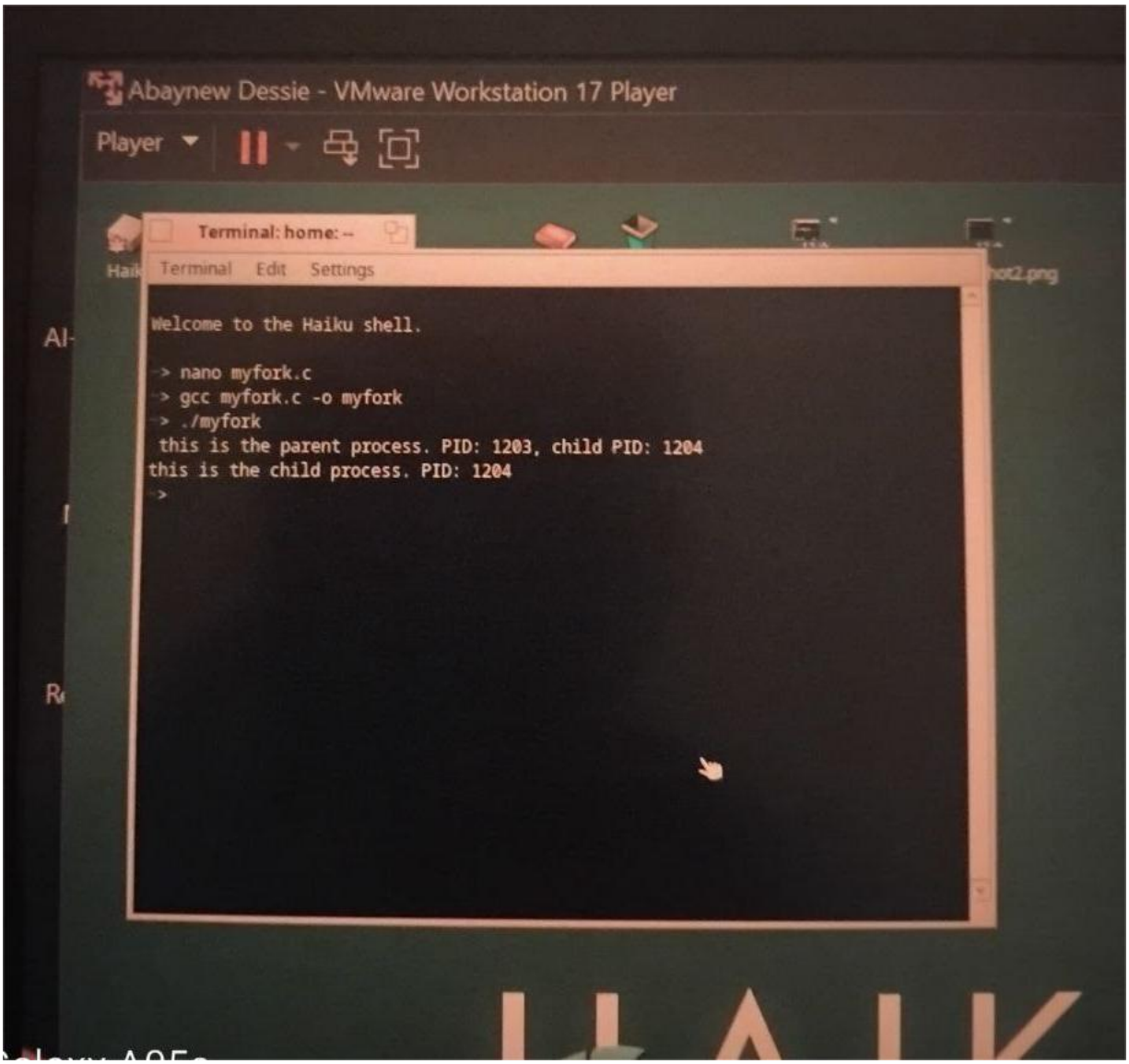
- ✓ Save the code as my`fork.cpp`.
- ✓ Open Haiku Terminal.
- ✓ Compile it:

```
g++ myfork.cpp -o myfork
```

- ✓ Run the program:

```
./myfork
```

Expected Output:



You will see two lines printed:

- this is the parent process
- This is the child process

Example:

```
This is the parent process! Child PID: 1203
This is the child process! PID: 1204
```

Note: Order may vary due to scheduling.

How `fork()` Works in the OS Kernel:

- The Haiku kernel handles `fork()` by:
  - Allocating a new process ID.
  - Copying the process memory space (Copy-On-Write optimization may apply).
  - Duplicating file descriptors and CPU registers.
- The child starts executing exactly where the parent left off—right after the `fork()` call.

Use Cases:

- Launching new programs (`fork()` + `exec()`).
- Creating daemon/background processes.
- Simulating parallel processing for practice.

Conclusion:

The `fork()` system call in Haiku OS provides a hands-on way to learn how processes are created and managed by the operating system. It is a gateway into understanding multitasking, inter-process communication, and kernel-level resource management.