BAHIR DAR UNIVERSITY

BAHIR DAR INSTITUTE OF TECHNOLOGY

FACULTY OF COMPUTING



**SUBMMIRION DATE:16/08/2017EC.**

**SUBMITTED TO: mr wendimu.**

OSSP project

Department: software engineering

Year: 2nd Year, 2017E.C

Project-work

Operating system and system programming

Prepaired by: Getahun Abe**be**

CONTENT COVERAGE

* INTRODUCTION
* BACKGROUND
* MOTIVATION
* OBJECTIVES
* KEY FEATURES OF FUCHSIA OS
* REQUIREMENTS
* PROBLEM FACED
* SOLUTIONS
* IMPLEMENTATION OF SYSTEM CALLS
* STEPS TO DOWNLOAD FUCHSIA OS
* ADVANTAGES AND DISADVANTAGES OF FUCHSIA OS

Introduction background of fuchsia OS on vmware workstation

Fuchsia OS is an open-source operating system developed by Google, designed to support a wide range of devices, from smartphones to IoT devices. It is built on a unique microkernel called Zircon, which emphasizes security, modularity, and scalability.

Background:

* The Evolving Landscape of Operating Systems: Traditional operating systems, while powerful, have faced challenges in adapting to the rapid advancements in hardware and software. Security vulnerabilities, monolithic architectures hindering modularity and updates, and limitations in real-time performance have become increasingly problematic, particularly in emerging domains like IoT and embedded systems.
* Lessons Learned from Android and Chrome OS: Google's experience with Android and Chrome OS, while successful, highlighted areas for improvement. Android's reliance on the Linux kernel has presented fragmentation and security update challenges. Chrome OS, while more streamlined, is still tied to the Linux kernel. Fuchsia is an opportunity to build a new OS from the ground up, incorporating best practices and addressing the shortcomings of existing systems.
* The Rise of IoT and Embedded Devices: The proliferation of internet-connected devices, from smart home appliances to industrial sensors, demands a new breed of operating systems. These devices often have resource constraints, strict real-time requirements, and stringent security needs. Existing OSes are often too heavyweight or lack the necessary security features to effectively manage these devices.

motivation

* Enhanced Security: Fuchsia's capability-based security model is a core motivation. By granting components only the necessary permissions through unforgeable capabilities, Fuchsia aims to minimize the attack surface and limit the impact of potential security breaches. This represents a significant improvement over permission models in other OSes, where vulnerabilities can often grant widespread access.
* Modularity and Updatability: Fuchsia is designed with a microkernel architecture, separating core system services from user-level applications. This modularity allows for easier updates, as individual components can be updated independently without requiring a full system reboot. This is crucial for maintaining security and stability over time.
* Real-Time Performance: Fuchsia's Zircon kernel is designed to provide predictable real-time performance, making it suitable for applications that require low latency and deterministic behavior. This is essential for embedded systems, robotics, and other applications where timing is critical.
* Versatility and Scalability: Fuchsia is intended to run on a wide range of devices, from embedded systems and smartphones to laptops and beyond. Its modular design allows it to be tailored to specific hardware configurations and use cases. It also aims to be scalable to accommodate future advancements in hardware and software.
* A Unified Platform: Fuchsia is envisioned as a potential unifying platform across various Google products, potentially replacing or complementing Android and Chrome OS in the long term. This could simplify development, improve user experience consistency, and reduce fragmentation.
* Innovation in OS Design: Beyond specific features, Fuchsia is driven by the general motivation to explore innovative OS design principles, investigate new programming languages (like Rust and Dart) within the OS ecosystem, and establish a foundation for future generations of computing devices.

objectives

1. Security:

* Enhanced Security Posture: To provide a significantly more secure operating system compared to existing solutions by employing a capability-based security model at its core. This minimizes the attack surface, restricts component access to only necessary resources, and reduces the impact of potential security breaches.
* Fine-Grained Permissions: To enforce fine-grained permissions and access control through capabilities, allowing for precise management of resources and minimizing the risk of privilege escalation.
* Robust Isolation: To ensure robust isolation between components, preventing malicious or compromised components from accessing sensitive data or interfering with other parts of the system.

2. Modularity and Updatability:

* Modular Architecture: To design a highly modular operating system based on a microkernel architecture, separating core system services from user-level applications.
* Independent Updates: To enable independent updates of individual components without requiring a full system reboot, minimizing downtime and ensuring timely security patches.
* Dynamic Composition: To support dynamic composition of components, allowing for flexible configuration and customization of the operating system for specific devices and use cases.

3. Real-Time Performance:

* Predictable Latency: To provide predictable and low-latency real-time performance for applications that require deterministic behavior, such as embedded systems, robotics, and industrial control systems.
* Resource Management: To implement efficient resource management mechanisms that guarantee timely allocation of CPU, memory, and other resources to real-time tasks.
* Responsiveness: To maintain responsiveness and stability even under heavy load, ensuring a smooth user experience for interactive applications.

4. Versatility and Scalability:

* Cross-Platform Compatibility: To design an operating system that can run on a wide range of devices, from embedded systems and smartphones to laptops and desktop computers.
* Adaptable Architecture: To create an adaptable architecture that can be tailored to specific hardware configurations and use cases, maximizing performance and efficiency.
* Scalable Design: To ensure that the operating system can scale to accommodate future advancements in hardware and software, supporting new technologies and emerging applications.

5. Ecosystem Development:

* Developer-Friendly Environment: To provide a developer-friendly environment with comprehensive tools, libraries, and documentation, enabling developers to create innovative applications for Fuchsia OS.
* Open Source Contribution: To encourage open-source contributions from the community, fostering innovation and collaboration in the development of the operating system.
* Partnerships: To establish partnerships with hardware manufacturers, software vendors, and other stakeholders to build a thriving ecosystem around Fuchsia OS.

6. Innovation and Research:

* Exploration of New OS Paradigms: To explore innovative operating system design paradigms, such as capability-based security, microkernel architecture, and component-based development.
* Language Exploration: To investigate the use of modern programming languages like Rust and Dart within the OS ecosystem, improving security, performance, and developer productivity.
* Future-Proofing: To create a foundation for future generations of computing devices, anticipating and addressing the challenges and opportunities of emerging technologies.

Key Features of Fuchsia OS:

**Zircon Microkernel**: Unlike monolithic kernels (e.g., Linux), Zircon is lightweight and designed for better security, modularity, and real-time performance.

**Modular Framework**: Applications and system components are built using a framework called "components," which makes updates and system maintenance easier.

**Cross-Platform Capabilities**: Designed to work seamlessly across a variety of devices, from mobile phones to smart home appliances.

**Modern Programming**: Supports Dart, the language used in Flutter, and allows developers to create high-performance apps with native-like experiences.

**Enhanced Security**: Implements modern security practices, including sandboxing and strict privilege separation, for a safer user experience.

**User-Centric Design**: Fuchsia experiments with innovative user interfaces, such as the "Armadillo" UI for touch-based devices.

# hardware and software requirments

To run Fuchsia OS on VMware Workstation, you need to meet specific hardware and software requirements. Here's what you'll need:

Hardware Requirements:

**Processor**: A modern 64-bit processor with virtualization support (Intel VT-x/AMDV).

**Memory (RAM)**: At least 8 GB of system RAM is recommended for smooth operation.

**Disk Space**: A minimum of 50 GB of free disk space to accommodate the virtual machine and Fuchsia OS image files.

**Graphics**: A GPU that supports OpenGL 3.3 or higher for better display performance.

**BIOS/UEFI**: Virtualization (Intel VT-x/AMD-V) must be enabled in your BIOS/UEFI settings.

Software Requirements:

**Operating System**: VMware Workstation runs on modern versions of Windows or Linux.

**VMware Workstation**: Ensure you have VMware Workstation Pro or Player installed (latest version recommended for compatibility).

**Fuchsia OS Build**: Either pre-built images from Google's repository or build it yourself from Fuchsia's source code.

**Dependencies**: If building Fuchsia OS, you'll need tools like Fuchsia SDK, GN build system, Clang, and other build dependencies, which are listed in the Fuchsia documentation.

**Here’s a step-by-step guide to installing Fuchsia OS on VMware Workstation:**

## Step 1: Prepare Your Environment

1. Enable Virtualization

- Access your computer’s BIOS/UEFI settings and enable Intel VT-x or AMD-V for virtualization.

2. Install VMware Workstation

- Download and install VMware Workstation Pro or Player (latest version recommended).

3. Download Fuchsia OS Image

- Obtain the precompiled Fuchsia OS image or build it from the Fuchsia source repository (if you prefer).

## Step 2: Create a Virtual Machine

1. Open VMware Workstation

- Launch VMware Workstation and click "Create a New Virtual Machine."

2. Choose Configuration

- Select "Custom" for advanced configuration options.

3. Select Guest OS

- Choose “Other” for the guest operating system, as Fuchsia OS is not listed in the predefined options.

4. Specify VM Resources

* Allocate CPU, RAM, and disk space based on your hardware capabilities:
* CPU: Minimum 2 cores (4 recommended).
* RAM: At least 4 GB (8 GB preferred).
* Disk Space: Allocate around 20–50 GB for the virtual machine.  Step 3: Attach Fuchsia Image

1. Attach Disk/File

- Add the Fuchsia OS image as the virtual hard disk or bootable media for the VM.

2. Configure Boot Options

- Set the VM to boot from the attached Fuchsia image.

Step 4: Boot and Run Fuchsia OS

1. Power On the VM

- Start the virtual machine in VMware Workstation.

2. Explore Fuchsia OS

- Once it boots up, experiment with the user interface, system commands, and features.

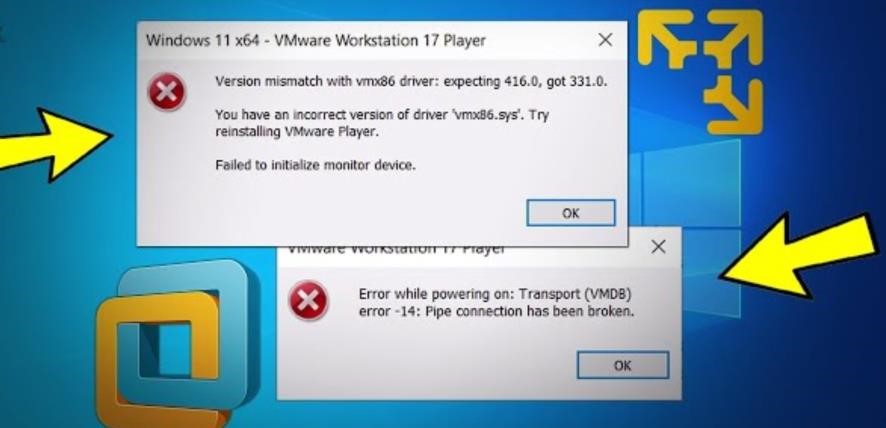
## Step 5: Troubleshooting (Optional)

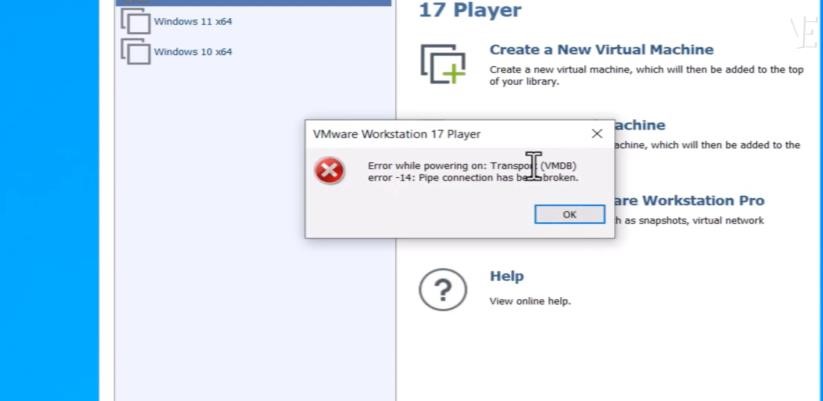
If you encounter issues during setup:

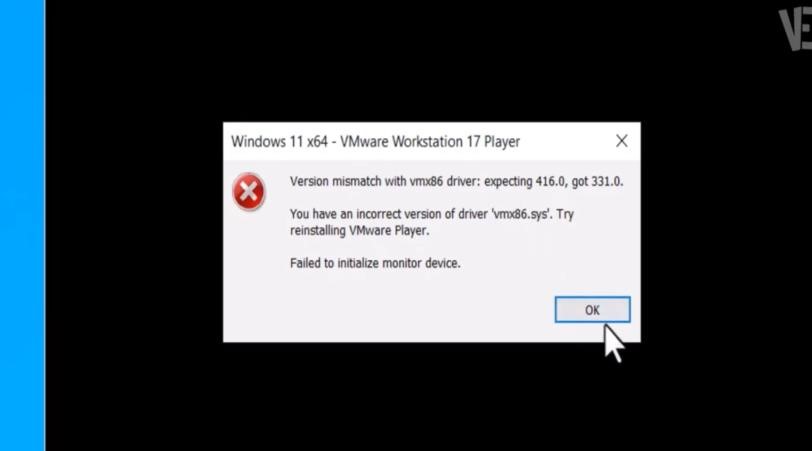
* Ensure your hardware supports virtualization.
* Verify the integrity of the downloaded Fuchsia OS image.
* Adjust VMware settings to match your system's configuration.

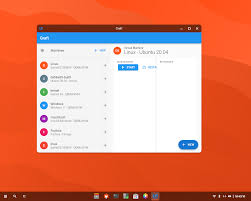
Facing challenges during the installation of Fuchsia OS on VMware Workstation is common, given its experimental nature. Here are some potential problems and their solutions: Common Problems and Solutions

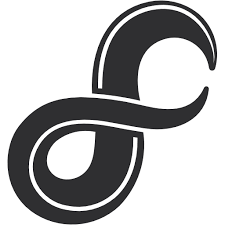






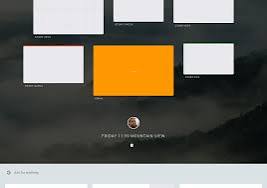


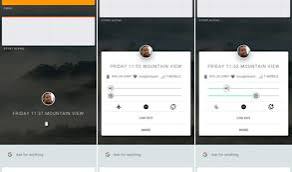


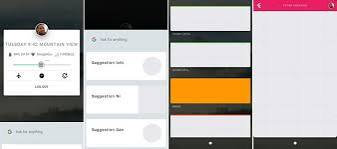


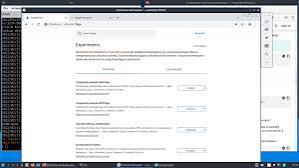


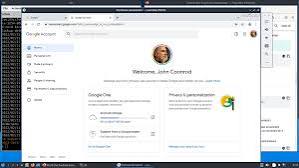
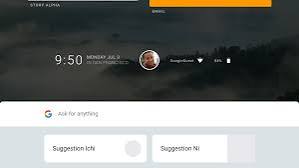


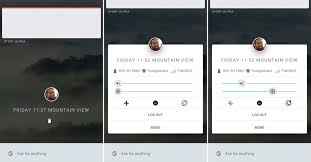












1. Virtualization Not Enabled

**Problem**: VMware displays an error stating that virtualization is not supported.

**Solution**: Check your BIOS/UEFI settings and ensure virtualization (Intel VTx/AMD-V) is enabled.

1. Fuchsia Image Doesn’t Boot

**Problem**: The virtual machine fails to boot or stops at a blank screen.

**Solution**: Verify the integrity of the Fuchsia OS image file. If you built the image yourself, ensure all build dependencies were correctly installed and the process completed successfully.

1. Inadequate Resources

**Problem**: The virtual machine runs slowly or crashes.

**Solution**: Increase CPU cores and RAM allocation in VMware settings. Fuchsia OS requires at least 4 GB of RAM (8 GB recommended) and 2 CPU cores (4 recommended).

1. Incorrect Disk Configuration

**Problem**: VMware cannot recognize the virtual disk containing Fuchsia OS.

**Solution**: Confirm the disk format (e.g., VMDK or ISO) is compatible with VMware. Ensure the disk is correctly attached in the virtual machine settings.

1. Compatibility Issues

**Problem**: VMware Workstation is outdated or lacks support for newer hardware features.

**Solution**: Upgrade VMware Workstation to the latest version to resolve compatibility issues.

1. Networking Issues

**Problem**: Fuchsia OS cannot access network features in VMware.

**Solution**: Configure the virtual network settings in VMware to use "NAT" or "Bridged" modes.

**Additional Tips**

* Consult Logs: Check the VMware log files for specific error messages.
* Explore Documentation: Refer to Fuchsia's [official installation guide](https://fuchsia.dev) for additional insights.
* Experiment: Adjust VMware settings gradually to resolve unique configurations or errors.

Fuchsia OS introduces a unique and innovative file system architecture that diverges from traditional systems like ext4 or NTFS. Here’s a breakdown of the file systems it supports and the rationale behind them:

### 1. Blobfsb

**Type**: A specialized file system for storing immutable data blobs.

**Purpose**: Blobfs is optimized for package-based data storage, allowing Fuchsia to manage application packages as immutable blobs.

**Why?** This approach ensures improved integrity, security, and reduced complexity in managing app packages.

### 2. Minfs

**Type**: A simple, high-performance file system for general-purpose use.

**Purpose**: Minfs is primarily used for file storage on devices running Fuchsia OS.

**Why?** Minfs emphasizes simplicity and efficiency, tailored for Fuchsia's modular and scalable design.

### 3. Fxfs

**Type**: A next-generation file system being developed for Fuchsia OS.

**Purpose**: Fxfs aims to address limitations of existing file systems and provide advanced features such as snapshots, compression, and improved reliability.

**Why?** Fxfs aligns with Fuchsia's goals of scalability, enhanced performance, and future-proof design.

•

### Why These File Systems Matter

Fuchsia’s file systems are designed to optimize the modular, scalable, and secure nature of the OS. By focusing on tailored file systems like Blobfs and Fxfs, Fuchsia can ensure efficient resource management, reliable package storage, and adaptability across diverse hardware platforms.

### Advantages

**Scalability Across Devices**:

Designed to work seamlessly on diverse hardware platforms, from smartphones to IoT devices, making it highly versatile.

**Enhanced Security**:

Built on the Zircon microkernel, which offers improved security by isolating system components and minimizing attack surfaces.

**Modular Architecture**:

Allows for easier updates, maintenance, and customization of system components without affecting the entire OS.

**Performance Optimization**:

Fuchsia is designed for efficient resource management, improving speed and battery life on devices.

**Future-Proofing**:

Incorporates modern technologies and programming frameworks, making it adaptable for future innovations.

**Unified Ecosystem**:

Aims to create a cohesive platform where applications and services can function smoothly across devices without compatibility issues.

**Open Source**:

Encourages collaboration and contributions from the global developer community, fostering innovation and transparency.

### Disadvantages

**Experimental State**:

Still under active development, which means it's not yet ready for widespread use or production environments

**Limited Ecosystem**:

Compared to established operating systems like Android or Windows, Fuchsia lacks a robust app ecosystem and user community.

**Hardware Requirements**:

Some devices may not fully support Fuchsia OS due to its unique architecture and resource demands.

**Compatibility Challenges**:

Limited support for legacy hardware and software, making adoption difficult for older systems.

**Learning Curve**:

Developers and users may face a learning curve due to Fuchsia’s novel design and different development frameworks (e.g., using Flutter and Dart).

**Uncertain Future**:

As Fuchsia is still an emerging technology, its long-term viability and adoption remain unclear.

In conclusion, Fuchsia OS represents a bold step forward in the world of operating systems. Its scalable, modular, and secure design, driven by the Zircon microkernel, sets it apart from traditional OS frameworks. By prioritizing future-proof technologies, enhanced performance, and a unified ecosystem, Fuchsia showcases Google's vision for adaptable and innovative platforms that can power the next generation of devices.

However, it’s important to acknowledge the challenges Fuchsia faces, such as its experimental status, limited ecosystem, and compatibility issues. As it continues to evolve, its potential impact on diverse industries—from IoT to mobile computing—is undeniable.

For developers, tech enthusiasts, and innovators, exploring Fuchsia OS provides an opportunity to contribute to and understand cutting-edge technology. Whether you're experimenting with it on VMware Workstation or diving into its unique file systems and frameworks, Fuchsia opens doors to a future shaped by modularity, security, and versatility.

### Future Outlook for Fuchsia OS

Fuchsia OS has a promising future as it continues to evolve and address the unique challenges faced by modern operating systems. Its modularity, scalability, and emphasis on security align perfectly with the growing demand for versatile platforms that can seamlessly power diverse devices. The focus on adaptability across IoT, mobile computing, and larger systems positions Fuchsia to potentially redefine the digital ecosystem.

As developers contribute to its open-source framework and Google refines its functionality, the following advancements can be expected:

**Expanded Ecosystem**: With active developer engagement, Fuchsia could establish a robust app ecosystem to rival Android and other OS platforms.

**Improved Device Integration**: Enhancements in cross-platform support may enable seamless interaction between various devices, creating a unified technological experience.

**Scalable Applications**: Fuchsia’s modular approach opens doors for scalable applications that can operate efficiently across resource-constrained and highperformance devices.

**Innovative User Interfaces**: The development of innovative user interfaces tailored for touch, voice, and gesture-based interactions could make Fuchsia stand out.

### Recommendations

**For Developers**:

**Learn & Contribute**: Explore Fuchsia’s documentation and contribute to its development to help shape its capabilities.

**Experiment with Flutter**: Since Fuchsia supports Flutter, mastering this framework will allow you to create optimized apps for future platforms.

**For Businesses**:

**Plan for Integration**: Businesses should monitor Fuchsia’s progress and start exploring how their products could benefit from its scalable, modular design.

**Adopt Early**: Tech enterprises could embrace Fuchsia for IoT and next-gen device applications to stay ahead of the curve.

**For Enthusiasts**:

**Stay Updated**: Follow Fuchsia-related news and updates to understand its ongoing development and impact on the technology landscape.

**Experiment**: Set up Fuchsia OS in virtual environments to explore its features and gain insight into its innovative design.

## Virtualization in Modern Operating Systems: What, Why, and How

What:

Virtualization is the creation of a virtual (rather than actual) version of computing resources, such as operating systems, servers, storage devices, or networks. In the context of modern OS, it typically refers to running multiple operating systems on a single physical machine using software called a hypervisor or virtual machine monitor (VMM).

Why:

Resource Optimization: Maximizes hardware usage by running multiple virtual machines (VMs) on a single physical system.

Isolation: Each VM runs independently, improving security and fault isolation.

Flexibility & Testing: Allows developers to test software on different OS environments without needing multiple physical devices.

Cost Efficiency: Reduces hardware costs and energy consumption.

Scalability: Enables easier deployment and scaling of applications in cloud environments.

How:

A hypervisor (like VMware Workstation, VirtualBox, or KVM) is installed on a host machine.

The hypervisor allocates hardware resources (CPU, memory, disk) to multiple VMs.

Each VM runs its own guest OS, independently from the host and other VMs.

The virtualization layer ensures isolation and resource management between all VMs.