[illegible]
Bahir Dar Institute of Tech
[illegible]
Bahir Dar University

OSSP Individual Project - BeOS OS

Name: Yonas Abate

ID: 1602838

Section: B

Department: Software Engineering

Faculty: Faculty of Computing, BDU

Submitted to Instructor:Wondimu Baye

Submission Date:16/08/2017 E.C

1. Introduction (Background & Motivation)

- *BeOS was developed by Be, Inc. in the mid-1990s and was designed to excel in multimedia performance, especially in the areas of audio, video, and graphics. It was highly regarded for its ability to handle concurrent tasks, such as audio and video processing, more efficiently than other contemporary operating systems. However, despite its performance strengths, BeOS was discontinued in the early 2000s due to market forces and lack of hardware support, as well as competition from other OS.*

2. Objectives

The primary objectives of this project are as follows:

1. *To install BeOS in a virtualized environment (such as VirtualBox or VMware) and understand its system architecture.*
2. *To explore the installation steps, analyze potential challenges, and investigate possible solutions to common issues that arise during installation.*
3. *To analyze BeOS's file system support, advantages, and disadvantages in the context of modern operating systems.*
4. *To simulate system calls and understand how BeOS and other OSes implement them to perform critical system-level tasks.*

3. Requirements

i. Hardware

- *The hardware requirements for running BeOS in a virtualized environment are relatively minimal, given the lightweight nature of the OS. However, since BeOS was designed before modern hardware specifications, certain compromises must be made to run it on current systems. These include:*

  - *RAM: At least 512MB (1GB recommended for smoother operation).*
  - *Processor: x86-compatible CPU.*
  - *Storage: A minimum of 2GB for the virtual disk, though more is recommended for additional space.*
  - *A virtualized CPU setup using software like VirtualBox or VMware Workstation.*

ii. Software

- *To run BeOS in a virtualized environment, the following software tools are required:*
  - *Oracle VirtualBox or VMware Workstation, both of which are capable of running older operating systems.*
  - *BeOS 5 Personal Edition ISO file, available from various archival sites.*

4. Installation Steps

Given the dated nature of BeOS, installation can be challenging on modern hardware. Below are the theoretical steps for installing BeOS in a virtualized environment like VirtualBox. If actual installation fails, this provides a guideline to simulate the process:

1. *Download the BeOS 5 Personal Edition ISO from an archive site like archive.org.*
2. *Create a new VM in VirtualBox with the following settings:*
   • OS Type: Other
   • RAM: 512MB
   • Virtual Disk: VDI format, with at least 2GB storage.
3. *. Mount the BeOS ISO to the VM's optical drive.*
4. *. Boot the VM — BeOS will load its graphical user interface (GUI).*
5. *. During installation, create a user account named "Yonas Abate" as required.*
6. *. Proceed with the BeOS installation, following the on-screen instructions. BeOS will automatically partition the virtual hard drive and begin installation.*
7. *. Once installed, explore the system to confirm proper installation.*

5. Issues (Problems Faced)

During the installation and testing phases, several common issues may arise when attempting to install BeOS in a virtualized environment:

1. ***Incompatibility with Modern CPUs**: BeOS was designed before the introduction of modern multi-core and 64-bit processors, making it incompatible with some hardware.*
   • Solution: Use an emulator like QEMU or try older versions of VirtualBox to mimic legacy hardware configurations.

2. *. **Lack of USB or SATA support**: BeOS doesn't support modern disk interfaces like USB or SATA, limiting the installation and hardware compatibility.*
   • Solution: Use IDE or legacy disk images for storage during the BeOS installation.

3. *. **Boot Failures Due to UEFI**: BeOS doesn't recognize modern UEFI boot configurations and may not start properly on newer systems.*
   • Solution: Configure VirtualBox/VMware for legacy BIOS booting instead of UEFI.

4. *. **No Active Updates or Patches** : BeOS is no longer maintained, so there are no security patches or software updates available.*
   • Solution: Consider using Haiku OS, an open-source BeOS-inspired OS that is actively maintained.

6. Filesystem Support

BeOS primarily uses its own file system, the Be File System (BFS), which offers advantages and limitations when compared to other common file systems.

- ***Be File System (BFS)**: BFS supports advanced features such as indexing, journaling, and high-performance data handling, which made BeOS particularly well-suited for multimedia applications.*

  *• **NTFS, FAT32, ext4, ZFS**: These file systems are not natively supported by BeOS. However, modern operating systems like Linux support ext4, while Windows uses NTFS and FAT32.*

  *• **BFS vs. Others:** Unlike most file systems, BFS's integration with BeOS allowed it to handle large volumes of media files, making it especially effective in multimedia applications.*

7. Advantages and Disadvantages

BeOS was an innovative system for its time, designed to provide efficient threading and high multimedia performance. However, its lack of compatibility with modern hardware and absence of updates make it impractical for current use.

Advantages:

• Excellent handling of multimedia tasks (audio, video, graphics).
• Efficient use of resources (e.g., CPU threads).
• Clean and user-friendly interface.
• Lightweight system suitable for low-end hardware.

Disadvantages:

• Lack of modern hardware support.
• No security patches or updates.
• No widespread adoption or third-party application support.
• Limited software ecosystem compared to Windows, macOS, and Linux.

8. Conclusion

- ➢ *BeOS represents a unique approach to operating system design, especially in terms of multimedia and threading. While it failed to gain mainstream adoption, it offers valuable lessons in resource management and performance optimization.*
- ➢ *The installation process for BeOS can be challenging on modern hardware due to lack of support for contemporary CPU architectures and peripherals. However, exploring its*

design principles and virtualized testing helps understand how early operating systems approached concurrent processing and multimedia handling.

9. Future Outlook / Recommendation

- *Given its historical significance, BeOS remains a valuable study in multimedia OS design, but its lack of modern hardware support and updates makes it impractical for real-world applications today.*
- *The Haiku OS project provides a modern alternative that mirrors BeOS's design while offering updated hardware support and community development. It is recommended that future research on BeOS-related technologies be done using Haiku OS, which continues the legacy of BeOS in a more accessible and supported manner.*

2. Virtualization in Modern Operating Systems

What is Virtualization?

- ***Virtualization** refers to the creation of virtual instances of physical resources, such as computing power, storage, or network functionality. This allows multiple operating systems or environments to run concurrently on the same hardware. Virtualization enables better resource utilization and isolation of tasks.*

Why Virtualization?

Virtualization is important in modern operating systems for several reasons:

- ***Isolation**: It allows different applications or services to run in isolated environments, which reduces the risk of cross-contamination (e.g., a failed service or app does not affect others).*
- ***Efficiency:** Virtualization helps maximize the use of hardware by running multiple virtual machines (VMs) on a single physical server.*
- ***Testing and Development**: It allows OS developers to test multiple versions of their systems or software on the same physical machine, thus enhancing productivity.*
- ***Resource Allocation:** Virtualization allows dynamic allocation of resources to different VMs based on their workload demands, improving system performance and scalability.*

How Virtualization Works in Modern Operating Systems?

- *Modern OSes support virtualization by using a **hypervisor**, which acts as a platform for managing and running virtual machines. There are two types of hypervisors:*
  - *Type 1 (Bare-metal): Installed directly on hardware (e.g., VMware ESXi, Microsoft Hyper-V).*
  - *• Type 2 (Hosted): Runs on top of a host operating system (e.g., Oracle*

VirtualBox, VMware Workstation).

- *Virtualization is implemented using hardware support (e.g., Intel VT-x, AMD-V) to allow the hypervisor to execute multiple VMs concurrently. The virtual machines run their own guest operating systems and can be managed and isolated independently of the host system.*

4. Implement System Calls

System Calls in Depth:

- *System calls act as the interface between user applications and the operating system's kernel. They allow user programs to request specific services that the kernel provides, such as interacting with the file system, memory, or network. This is crucial for enabling programs to execute tasks that require privileged access to system resources, such as reading from a file or creating a new process.*

A system call typically involves the following:

1. ***User Application**: The program that calls the system service.*
2. ***Library/Wrapper Function**: The function that invokes the system call on behalf of the user application. In many operating systems, these are implemented in C standard libraries (e.g., `glibc` in Linux).*
3. ***Kernel**: The core part of the OS that provides services such as file management, process scheduling, and memory management. The kernel handles the actual execution of the requested system service.*
4. ***System Call Interface**: This is the mechanism that switches the CPU from user mode to kernel mode, allowing the kernel to perform privileged operations.*

Types of System Calls:

1. ***Process Control**:*
   - ***fork()**: Creates a new process.*
   - ***exec()**: Replaces the current process with a new one.*
   - ***exit()**: Terminates the current process.*
2. ***File Manipulation**:*
   - ***open()**: Opens a file for reading or writing.*
   - ***read()**: Reads data from a file.*
   - ***write()**: Writes data to a file.*
   - ***close()**: Closes an open file descriptor.*
3. ***Memory Management**:*
   - ***mmap()**: Maps files or devices into memory.*
   - ***brk()**: Changes the data space of a process.*
4. ***Communication**:*
   - ***socket()**: Creates a socket for network communication.*

- ***bind()**: Assigns an address to a socket.*
- ***listen()**: Prepares a socket to accept incoming connections.*

Example: System Call Implementation in BeOS

In BeOS, system calls are used for tasks like interacting with files, processes, and network services. Below is a practical example of implementing system calls to create a file, write to it, and verify its existence.

BeOS System Call Example: File Manipulation

In BeOS, you can use C++ to make system calls for file operations. The BeOS API allows for easy file handling with methods like `open()`, `write()`, and `close()`.

```
cpp
#include <iostream>

#include <fstream>

#include <cstdio>

using namespace std;


int main() {

    ofstream file("newfile.txt");

    if (!file) {

        cerr << "File creation failed." << endl;

        return 1;

    }

    file << "This is a system call demonstration." << endl;

file.close();

FILE *checkFile = fopen("newfile.txt", "r");

    if (checkFile) {

        cout << "File created successfully." << endl;

        fclose(checkFile);
    } else {
        cerr << "File does not exist." << endl;
    }

    return 0;
}
```

Output: File created successfully.

explanation

- Since the file `newfile.txt` is successfully created and written to, the program checks its existence using `fopen()` in read mode.
- The file is found → `checkFile != NULL` → so it prints:
  "File created successfully."

Explanation:

ofstream*: This object is used to open a file for output. If the file does not exist, it will be created. The << operator writes the string into the file.*

fopen()*: A C standard library function used to open the file in read mode and check if it exists. If the file is opened successfully, it prints a success message. Otherwise, an error message is displayed.*

file.close()*: Closes the file to ensure that data is written and resources are freed.*

Note since BEOS OS is an outdated operating system installing and show snippted screenshot is not necessarly based on given instruction .These are my individual assignment thanks.