**BLOCKS OS**

**An Engineering Project in Community Service**

**Phase – II Report**

***Submitted by***

**Team Members List**

|  |  |  |
| --- | --- | --- |
| **SERIAL NUMBER** | **NAME** | **REGISTRATION NUMBER** |
| **1** | **Pavir Kadain** | **19BCE10006** |
| **2** | **Abhishek Srivastava** | **19BCE10071** |
| **3** | **V Surya Kumar** | **19BCE10286** |
| **4** | **Saransh Pratap Singh** | **19BCY10035** |
| **5** | **Pratul Maurya** | **19BCY10036** |
| **6** | **Viplav Khubchandani** | **19BAI10106** |
| **7** | **Anjali Singh** | **19BCG10003** |
| **8** | **C S Soujanya Mudliar** | **19BCG10094** |

***in partial fulfillment of the requirements for the degree of***

***Bachlore of Engineering and Technology***



**VIT Bhopal University**

**Bhopal**

**Madhyapradhesh**

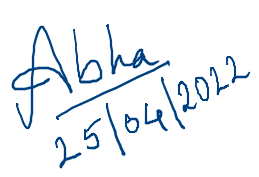
**21 April 2022**



**Bonafide Certificate**

Certified that this project report titled **“BLOCKS OS”** is the bonafide work of “19BCE10286 V Surya Kumar, 19BCE10071 Abhishek Srivastava, 19BCE10006 Pravir Kadian, 19BCG10003 Anjali Singh, 19BCY10036 Pratul Maurya, 19BCY10035 Saransh Pratap Singh, 19BAI10106 Viplav Khubchandani, 19BCG10094 C.S. Soujanya Mudliarwho carried out the project work under my supervision.

This project report (Phase II) is submitted for the Project Viva-Voce examination held on 21/04/2022



Dr. Abha Trivedi

**Supervisor**

**Comments & Signature (Reviewer 1)**

**Comments & Signature (Reviewer 2)**

1. **Introduction**

Open-source operating system Blocks-OS is designed to be used for learning and learning how an operating system performs. The user can experience first-hand how interrupt vectors, memory management, segmentation, and other concepts such as interrupts work through this project.

This project has several advantages over a higher-level project that isolates students from the machine:

Learning about segmentation, interrupt vectors, and memory management through hands-on experiences helps students understand the computer more deeply.

When forced to program without the familiar library functions, students discover just how much support operating systems provide.

When students are able to test their operating systems on their own machines, they make otherwise abstract concepts more tangible.

Branched development allows students to pick any branch and start implementing the project's rest of the features of any particular feature by using the project as boiler-plate code and detailed documentation as guidelines.

**1.1 Motivation**

People develop operating systems for a variety of reasons. Each individual developer may have their own, but some reasons are common among some developers:

Having complete control over the machine. When designing an application or other user space program, the developer should consider the code written by others: operating system, libraries, other programs, etc. It is a powerful feeling that only one code running on a machine will belong to you.

Research. A few operating system projects are started as homework or research projects. Research challenges are often created to improve existing operating systems.

Making low-level plans is a fun and exciting job because one has to do everything. This may seem like a lot of challenge but for the same reasons it is a lot of fun.

As software developers, we need to know how systems work, how they fit in, and the internal functioning of your system.

**1.2 Objective**

Current operating system curricula emphasise the theoretical side of the subject. By documenting each step, our project will help the student developer community learn about the practical aspects of the subject.

BLOCK OS is different from other pre-existing projects as existing projects are mostly incomplete, poorly documented, and with no community support. BLOCK OS is implemented in the following way:

* Our project is developed specifically for the student community to learn and follow the overall process of how an OS is developed from scratch, what goes behind the scenes of the working of operating systems, and a lot more.

* So, to develop an OS that any student can follow we need very adequate and sequenced documentation so anyone who reads it will be able to understand it easily.

* Documentation, as well as implementation, will be well defined and divided into modules.

1. **Existing Work**

You can find numerous works and projects about operating systems on the Internet. But to work on an OS-related project, you need to know what makes them unique, what they do, and what makes them follow a particular architecture.

The educational operating system implementation is NachOS. This is an example of an operating system developed by the developer community or developers. UNIX process is how it runs . Features such as threads, remote procedure calls, recent hardware advances such as RISC, and memory tier spikes are easy to implement. In addition, it implements a protocol, which is a common design technique.

In addition, there are a variety of modern software and design techniques such as protocol tiering and object-oriented programming.

Another example of an operating system is PintOS inspired by the NachOS Educational Operating System at the University of California, Berkeley. Like other operating systems, PintOS, written in C, is a simple operating system framework for the 80x86 architecture. It implements various features such as kernel thread support, loading and running user programs, filesystems, etc. at a very simple and basic level. PintOS can be easily run in the system simulator.

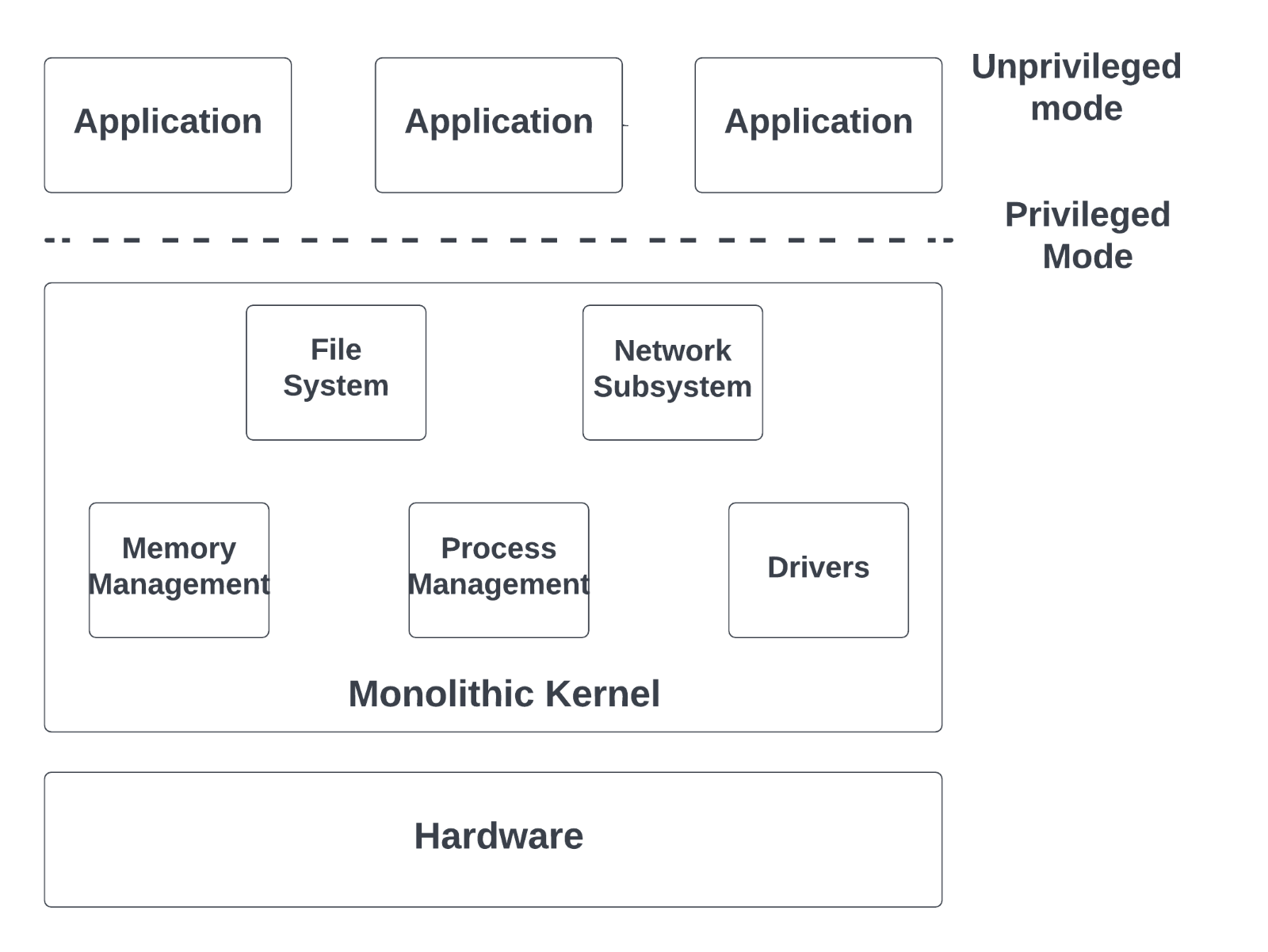
PintOS started as an alternative to NachOS, which shares strong design similarities. PintOS differs from NachOS in two important ways. First of all, PintOS uses 80x86 virtual or virtual hardware, while NachOS works as a processor in the host application. Second, unlike most real-life applications, PintOS is labeled C ++, while NachOS is labeled C. Wiki OS dev is a website that provides information about building an operating system and serves as a community for people interested in developing an operating system with 693 wiki articles. But when talking about a real implementation using it, it's a bit hard to rely on alone, as it's a condensed document that outlines 12 features at once.

There are many other examples of such implementations of different types of operating systems, but after reviewing and investigating some of them, creating an operating system with an original design derived from an existing operating system. I came up with the idea The implemented architecture is derived from some additional features aimed at learning and understanding

1. **Monolithic Operating System**
2. **System Design / Architecture**

An operating system that is monolithic controls everything in the kernel, including file management, memory management, device management, and processes management. This allows the kernel to access all resources assigned to the operating system. All elements of the software package reside inside the kernel in monolithic systems.

IPC is the means through which both kernel space and user space from a monolithic kernel communicate with each other.



**Fig 1 Monolithic operating system architecture diagram**

Features of the monolithic kernel -

* It has quick and strong code.
* All parts directly communicate with each other.
* It works well for smaller tasks.
* It has an easy structure and all parts are embedded in the kernel for the process.

Some of the advantages of the monolithic kernel are −.

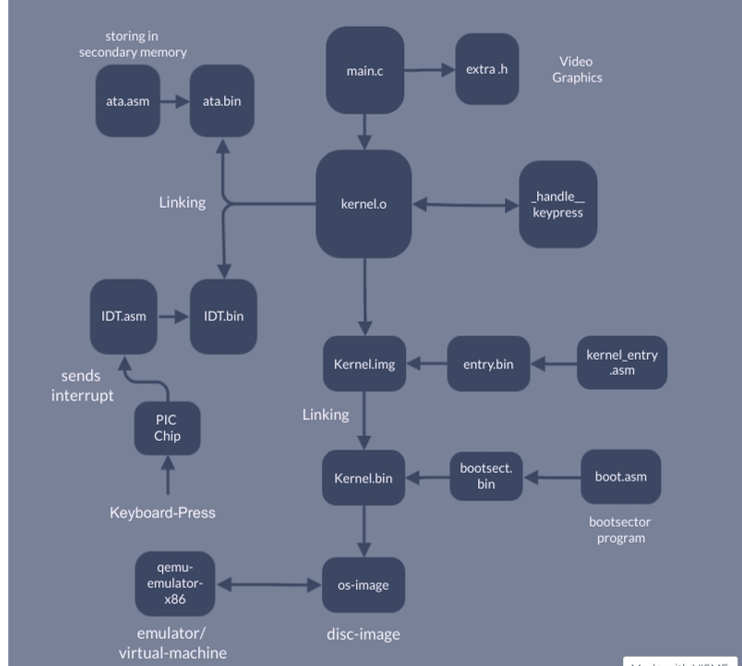
* In terms of execution speed, the monolithic kernel is relatively fast, as are the services such as memory management, file management, and schedulers; they are implemented under a common address space.
* The monolithic kernel runs every process in its own address space.
* Monolithic kernels are binary files stored in static locations.

1. **Working Principle**

It is the software of the operating system that makes it work, and works perfectly in compliance with the available resources and protocols. The process involves different aspects that are overseen, executed and governed by the specific software.

* Process management
* Memory management
* File management
* Device management
* I/O System management
* Secondary storage management
* Command interpretation
* Networking
* Job accounting

Block diagram of all the connected files of the project -

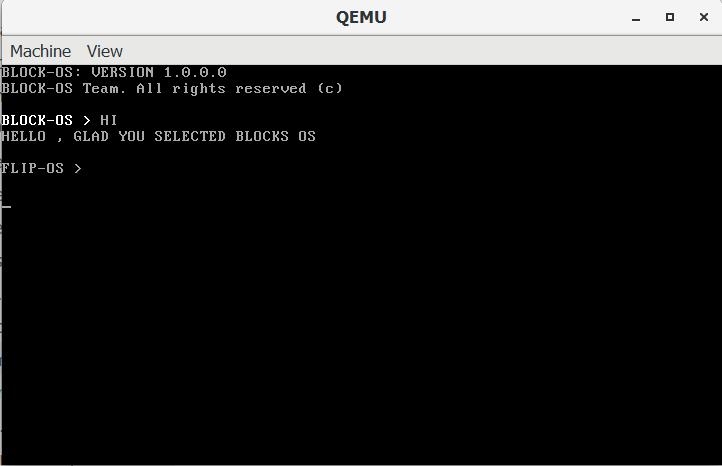
****

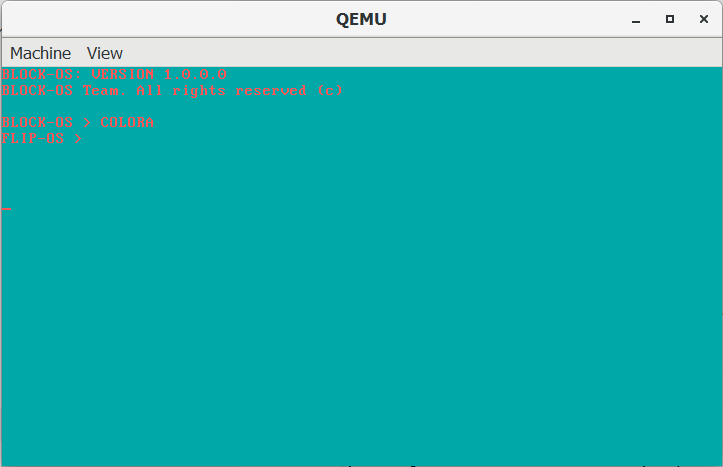
1. **Results and Discussion**

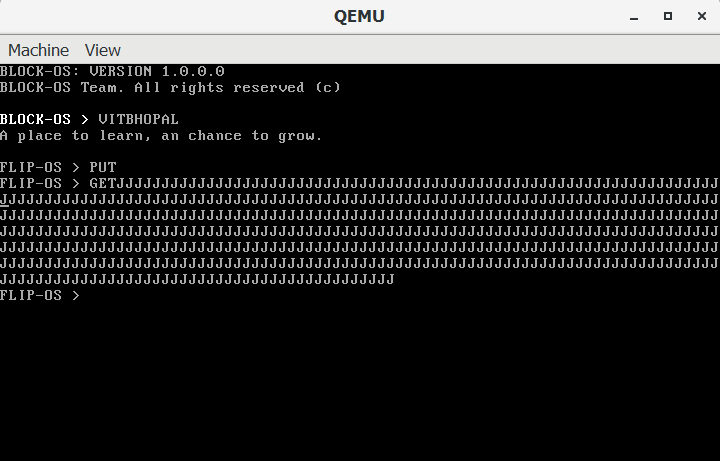
Blocks-OS is an operating system that is designed from scratch and open-source. Our goal was to demonstrate our Operating System as a learning tool for the students to get a deeper understanding on how OS works.

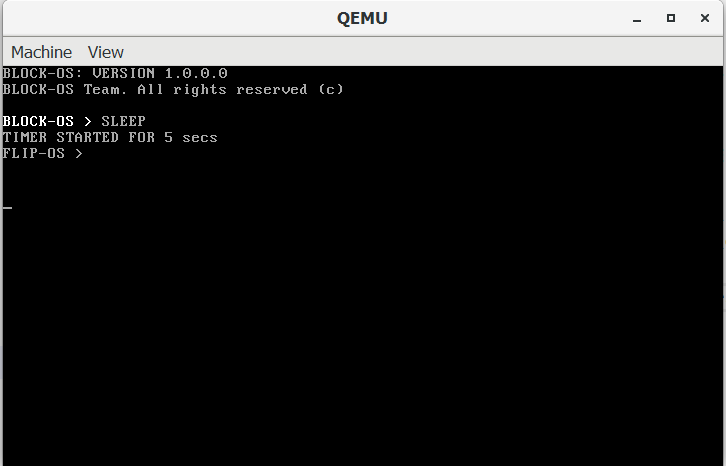
Using version control during the development control allows the students to learn from the development phase and a deeper understanding of the basics and core concepts of operating system development.

We have a cross-compiler set up for operating system development, and a basic kernel from which prints” Hello World” to the screen by writing characters directly to the video memory. The OS will have a kernel interrupt handler and system call handlers to print a string to the console and read a string from the keyboard. The OS will have a memory management system along with a rudimentary shell to read files from the disk and execute and terminate programs.









**e. Individual Contribution**

I implemented the interrupt descriptor table in the kernel using C, the data structure used by the x86 architecture. The Interrupt descriptor table is used by the processor to determine the correct response to interrupts and exceptions.

Use of the IDT is triggered by three types of events: hardware interrupts, software interrupts, and processor exceptions, which together are referred to as interrupts. I defined it for the hardware interrupts. I have also worked on the converter, which will be used to convert the scan codes from the keyboard to ASCII values that are readable.

1. **CONCLUSION**

The basic idea behind the Block OS project is to create an OS for the student community and to showcase how an OS is being developed step by step.

The main method that is followed while creating this Operating system is the software development waterfall model method, which involves a sequential development process that flows like a waterfall through all processes of a project (like, analysis, design, development, and testing).

For example, while analyzing, we saw the structures of the current existing OS like NachOS and ping OS to get the basic idea of the architecture and working of these OS and followed it developed our own.

Designing - A roadmap is created to follow at each and every step while developing.

Development - We made a basic startup program.

Testing - The testing of the basic programs created was done on Qemu. Virtual machines, cloud OS.

1. **Reference:**

* <https://wiki.osdev.org/Main_Page>
* <https://github.com/cfenollosa/os-tutorial>
* <https://drive.google.com/file/d/1bUAbfE7OU6NjnyFwVGGkeHR11BPq1l32/view>
* <https://www.cs.bham.ac.uk/~exr/lectures/opsys/10_11/lectures/os-dev.pdf>
* [CSE 221 - Graduate Operating Systems](https://cseweb.ucsd.edu/classes/sp00/cse221/projects.html)

**Plagiarism report**

<https://drive.google.com/drive/folders/1igGY4cr9Ak3uKTHPFhoywD33j8_6mxKx?usp=sharing>