# BLOCKS OS

# An Engineering Project in Community Service

## Phase – II Report

**Submitted by**

|  |  |  |
| --- | --- | --- |
| **SERIAL NO** | **NAME** | **REGISTRATION NUMBER** |
| 1 | V Surya Kumar | 19BCE10286 |
| 2 | Abhishek Srivastava | 19BCE10071 |
| 3 | Anjali Singh | 19BCG10003 |
| 4 | Pratul Maurya | 19BCY10036 |
| 5 | Pravir Kadian | 19BCE10006 |
| 6 | Saransh Pratap Singh | 19BCY10035 |
| 7 | Viplav Khubchandani | 19BAI10106 |
| 8 | C S Soujanya Mudliar | 19BCG10094 |

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

***Bachelor of Engineering and Technology***

****

**VIT Bhopal University**

**Bhopal**

**Madhya Pradesh**

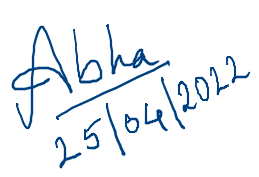
**21 April 2022**

****

**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 Mudliar** who carried out the project work under my supervision.

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



Dr. Abha Trivedi

**Supervisor**

**Comments & Signature (Reviewer 1)**

**Comments & Signature (Reviewer 2)**

**INDEX**

|  |  |  |
| --- | --- | --- |
| S.NO | TITLE | Page No |
| 1 | Introduction | 4 |
| 1.1 | Motivation | 5 |
| 1.2 | Objective | 6 |
| 2 | Literature Review | 7-8 |
| 3 | Topic of The work | 9-15 |
| 3.1 | System design/Architecture | 9-13 |
| 3.1 | Figure 1(monolithic structure) | 9 |
| 3.2 | Working Principle | 7-13 |
| 3.3 | Results and Discussions | 13-14 |
| 3.4 | Individual Contribution | 15 |
| 4 | Conclusion | 16 |
| 5 | Reference | 16 |

**INTRODUCTION**

An Operating System (OS) is a software that acts as an interface between computer hardware components and the user. Every computer system must have at least one operating system to run other programs.The OS helps you to communicate with the computer without knowing how to speak the computer’s language.

BLOCKS-OS,is an operating system we made, solely for educational purpose. We were able to understand the working principles of operating system.End-users can use this project to learn more about how computers work by experiencing first-hand how concepts such as interrupt-vectors, and memory management, segmentation actually manifest themselves on their own computers.

Throughout the project, we kept a well mannered documentation of the progress. It was done in a way such that even a beginner in the topic would be able to understand the basic concepts and also would be able to work on making their own operating system. This helps the students and learning community a lot since there aren’t many guides on how to build their own operating system out there.Students learn how much support they get from operating systems when forced to program without the familiar library functions.Branch-wise development helps students to pick any branch of the complete project and start implementing rest features or any particular feature by using the project as boiler-plate code and detailed documentation for reference.

By experiencing such concepts as segmentation, interrupt vectors, and memory management in action, students gain a deeper understanding of the computer itself.Students learn how much support they get from operating systems when forced to program without the familiar library functions.

* 1. **MOTIVATION**

While thinking on a topic for our project we had a lot of choices but we wanted to try something challenging, and thus came the idea of developing our own operating system. While we were thinking about how should we approach this idea, we realized that there aren’t any proper guides on how should any beginner should start working on making their own Operating System. Thus we took up on that and that became the main motivation for our project, to develop an operating system from scratch while documenting every steps to make an easy to understand guide for other students.Having complete control over the machine. When developing an application or other userspace program, the developer has to take the code written by others into consideration: the operating system, the libraries, other programs, etc. It is a powerful feeling for the only code to be running on a machine to be your own.Low-level programming is a fun and exciting task because one has to do everything. This may seem more challenging but for the same reasons more fun. As a software developer we should know how systems work, how it all fits, and the innermost workings of your program.

* 1. **OBJECTIVE**

Current operating system curricula emphasize 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.

**2) LITERATURE REVIEW/EXISTING WORK**

We can find numerous papers and projects on operating systems on the internet, but in order to work on any OS-related project, it is necessary to know what makes them unique, what they do, and what makes them follow the particular architecture.

ToaruOS is a "complete" operating system for x86-64 PCs and experimental support for ARMv8.The OS includes a kernel, bootloader, dynamic shared object linker, C standard library, its own composited windowing system, a dynamic bytecode-compiled programming language, advanced code editor, and dozens of other utilities and example applications.

Features

1) Dynamically linked userspace with shared libraries and dlopen.

2) Composited graphical UI with software acceleration and a late-2000s design inspiration.

3) VM integration for absolute mouse and automatic display sizing in VirtualBox and VMware Workstation.

4) Unix-like terminal interface including a feature-rich terminal emulator and several familiar utilities.

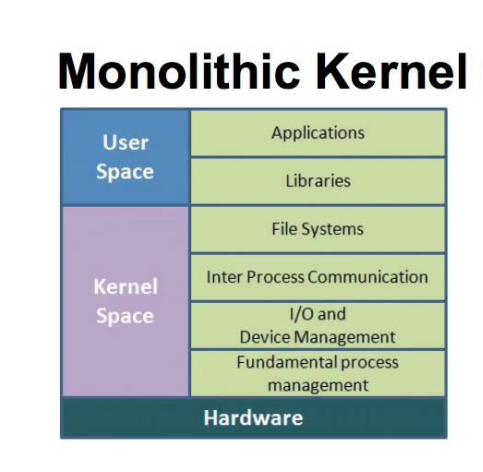
5) Optional third-party ports including GCC 10.3, Binutils, SDL1.2, Quake, and more.

One another example of an operating system is PintOS instructional operating system from the University of California, Berkeley. PintOS is written in C language much like other operating systems and is a simple operating system framework for the 80x86 architecture. It implements various functionalities like kernel threads support, loading and running user programs, and a file system at a very simple and basic level. PintOS can be easily run in a system simulator.

Wiki OS dev is a website that provides information about the creation of operating systems and serves as a community for those people interested in OS development with 693 wiki articles. But when we talk about actual implementation using it, it is a bit hard to only rely on this because it is more sort of compressed documentation with an overview of 1-2 functionalities of each.

**3) THE TOPIC OF WORK**

**3.1) SYSTEM DESIGN/ARCHITECTURE(Monolithic OS)**

The monolithic operating system is also known as the monolithic kernel. This is an old type of operating system. They are used to perform small tasks like batch processing, time sharing tasks in banks. Monolithic kernel acts as a virtual machine which controls all hardware parts. The monolithic operating system is a very basic operating system in which file management, memory management, device management, and process management is directly controlled within the kernel. All these components like file management, memory management etc. are located within the kernel

*Figure 1*

The kernel will access all the resources given within the system. In monolithic systems, every element of the software package is contained inside the kernel.​

Monolithic kernel has two parts kernel space and user space, both parts communicate with each other through IPC.​

Some of the advantages of monolithic kernel are −

The execution of the monolithic kernel is quite fast as the services such as memory management, file management, process scheduling etc.are implemented under the same address space.

A process runs completely in a single address space in the monolithic kernel.

The monolithic kernel is a static single binary file.

**3.2) WORKING PRINCIPLE**

It is the operating system's software that makes it operate, and it works flawlessly in accordance with the resources available and protocols. Different components of the process are overseen, executed, and managed by the specific software.

**Boot Sector**

The Boot Sector is the part of the hard drive where we put the programmes we want to run. It usually contains code to load the operating system that is installed on that machine. The code in the boot sector, however, is not the first to be performed.

When we turn on our computer, it begins to copy some code called Bios from the ROM (Read Only Memory) to the main memory (RAM). The CPU then performs it.

Boot sector by initialising first 510 byte of hard disk as 0 and then initialising last two bytes with value aa and 55 as shown in code below.

**Boot Loader**

The boot loader is a component of the boot sector that loads the remaining operating system components. ​

In a c file, we created a boot loader that loads 30 sectors into main memory.

We introduced the START() method in our C programme and developed a linker file that automatically calculates where the entry function is present to execute this C file.

**Audio**

A speaker can have either "in" and "out" positions. A value of 1 causes the speaker to move to the "out" position, whereas a value of 0 causes it to move to the "in" position. If the speed of repetition is within the range of what the speaker can emit and what the human ear can hear, audible tones are produced. ​

In the extra.h file, we have implemented three functions:

· void scream(int fr); // passes the frequency of the sound that will be played

· void play(unsigned int fr ); // plays the sound​

· void stop( ); //stops the sound

The audio is played when a user enters the PLAY() command.

### Video Graphics: Graphic Driver​

Using the graphic driver, we built a variety of printf commands and colour changing functions, including:

### Simple Video Player

Video is a term used to describe moving images that we perceive as videos. ​

To provide us with a video, the computer continuously changes the data stored in the video memory, giving us the impression that it is moving or giving us a sense of "Video."

Because our processor is so fast, it needs to keep the frame update rate low in order to keep up with the refresh rate of our monitor.

### Secondary Memory​

A hard disc, often known as a primary "secondary storage device," is a crucial component of a computer. Its capabilities are also utilized to maximize the usage of the primary storage device or RAM, in addition to storage. This is used in concepts like paging. We commonly use the in and out assembly commands when implementing a hard disc driver.

There are several methods for reading and writing to a hard disc. The LBA (Linear Block Address) mode is what we've employed. This is the simplest approach to read/write to a hard disc; all we need is the sector's Block address.

Passing 0 allows us to enter the first sector (The boot sector). Please note that writing to the 0th sector may cause your computer to become unbootable; however, you may always copy a boot loader to that sector. ​

Earlier techniques that exposed the physical details of the storage device to the operating system software were replaced by the LBA scheme.

The cylinder-head-sector (CHS) scheme was the most common, in which blocks were addressed using a tuple that identified the cylinder, head, and sector where they existed on the hard disc.

CHS addresses can be converted to LBA addresses using the following formula:​

LBA = (( C x HPC ) + H ) x SPT + S - 1​

where,​

C, H and S are the cylinder number, the head number, and the sector number​

LBA is the logical block address​

HPC is the number of heads per cylinder​

SPT is the number of sectors per track

We devised two instructions, GET and PUT, to make the hard disc work in ATA in LBA mode. The write method is called by the PUT commands, and it writes a character array to the Hard Disk. The read technique was used to retrieve the array of characters from the Hard Disk by the GET command. ​

When we input and enter the PUT command, it first copies the value 0 to blockAddr, then sets the value 'J' in every cell in the At[] character array, and lastly adds a null character.

### Keyboard Driver​

The CPU will call a function that we have defined whenever a key is pushed. The CPU will not handle keyboard logic, but it will allow us to run code whenever a key is pressed. ​

The CPU will call a function that we have defined whenever a key is pushed. The CPU will not handle keyboard logic, but it will allow us to run code whenever a key is pressed. ​

To handle keyboard input, we'll need to write some code, send its address to the interrupt descriptor table, and tell the processor to load it

It will call our keyboard handling code whenever a key is pressed after loading all of the parameters and the location of that code.

When this interrupt occurs, we can try reading from the keyboard to see which key has been pushed. ​

What keyboard gives as the value for pressed key is not ascii. It is named as scan codes. Modern keyboard drivers connect with the computer via USB interface, and while we are utilizing PS/2 keyboards, USB keyboards will also operate because they simulate the older PS/2 keyboard.

The PIC, or Programmable Interrupt Controller, is a computer chip that generates interrupts. When a key on the keyboard is pressed, the keyboard's chip instructs the computer's pic chip to issue an interrupt. ​

The photo chip will then determine when the cpu should be notified of the interrupt. When the cpu receives the message that a key is being pressed, it performs the code that we previously instructed the cpu to run when a key is pushed.

### Programmable Interval Timer​

This is used to generate delays in the OS code, which can then be used by other components in procedural statements for delay control. ​

An oscillator, a prescaler, and three separate frequency divisions make up the Programmable Interval Timer (PIT) chip (Intel 8253/8254). Each frequency divider includes an output that is used to control external circuitry by the timer.

The PIT chip's oscillator operates at 1.193182 MHz (approximately). The following I/O ports are used by the PIT chip

0x40, 0x41, 0x42, 0x43

The Timer IRQ works as follows: By using the PIT to generate a hardware interrupt every n milliseconds, we may make a basic timer. Decrement this variable until it reaches 0 every time the timer interrupt is called. Implementing the sleep function, which waits for the interval.

### Global Constructors

Global constructors are similar to those found on global C++ objects, and they are designed to run before the main function. ​

The constructors are in charge of reading command line arguments, initializing the standard library (memory allocation, signals, etc. ), performing global constructors, and eventually exiting (main(argc, argv)).

This is something that should be done before calling the main method.

​Invoking the global constructors/destructors is as easy as traversing the array and running each element.

## 3.3) Results and Discussion

We have successfully implemented the following on our operating system from scratch: Set the BIOS and kernel to their default settings. For the booting procedure, there is a Boot Sector and a Boot Loader. GDT implementation (Global descriptor table). Change from 16-bit to 32-bit protected mode. Support for C language has been added. A Graphic Driver was implemented to display graphics on the screen. Also a simple video player was implemented to better facilitate our OS. COLORA (for altering the color of the screen monitor), VITBHOPAL (for displaying messages on the screen), and other features have been added. ISR has been implemented (Interrupt Descriptor Table). Using ISR and PIC, create a keyboard driver. ATA is used to write and read data from hard discs.

The PLAY() command can be used to provide audio capability. Basic arithmetic operations. Paging on IA-32 architecture. Programmable Interval Timer

Students can learn from the development phase and gain a greater understanding of the fundamentals and fundamental principles of operating system development by using version control during development and the documentation we developed recording the development phases.

**3.4) INDIVIDUAL CONTRIBUTION**

I have done the documentation of GDT which define the characteristics of the various memory areas used during program execution, IDT which is used by the processor to determine the correct response to interrupts and exceptions, ISR which is a software process invoked by an interrupt request from a hardware device and it handles the request and sends it to the CPU, interrupting the active process ,Memory Implementation (HDD and SDD), calling Global Constructors which has the responsibility of parsing the command line arguments, initializing the standard library, Programmable Interval Timer(PIT) which is a counter that generates an output signal when it reaches a programmed count , PAGING on IA-32 Architecture which allows each process to see a full virtual address space, without actually requiring the full amount of physical memory to be available or present. This was one of our major objective of our project since our aim was to create a documentation which we can present that will help other students to learn the basics of developing operating system and there aren’t many guides out there that can help students to develop operating system from scratch. I have also worked on literature review and survey of other people who took the project of building their own operating system and had to look for what were the major points of differences in them and how we could work on making our own unique operating system.

4) CONCLUSION

We have developed our own open source operating system -BLOCKS OS from scratch, which is based on monolithic architecture. We developed the operating system based on software development waterfall model method. The OS contains the basic functions and is created as a learning tool for the students to get a deeper understanding on how Operating System works through the documentation done.

5) REFERENCES

5.1)https://wiki.osdev.org/Main\_Page

5.2)https://www.youtube.com/watch?v=YbXHU7W7Its

5.3)writing an operating system - Nick Blundell

<https://www.cs.bham.ac.uk/~exr/lectures/opsys/10_11/lectures/os-dev.pdf>

5.4)https://en.wikipedia.org/wiki/Global\_Descriptor\_Table

5.5)https://wiki.osdev.org/Programmable\_Interval\_Timer#The\_Oscillator

5.6)https://wiki.osdev.org/ATA\_PIO\_Mode

5.7)https://wiki.osdev.org/Category:ATA

5.8)<https://drive.google.com/file/d/1qviDBcRat4fuSdgx_IYXii2_ZFFmZK8Y/view?usp=sharing>

5.9)<https://web.stanford.edu/class/cs140/projects/pintos/pintos_1.html>

5.10)<https://cseweb.ucsd.edu/classes/sp09/cse120/projects/nachos.pdf>

**6) Plagiarism Report**

https://drive.google.com/drive/folders/1dO-YYuOeseKoCb7LyR7gHreYoYZqXRNa