শিক্ষা নিয়ে গড়ব দেশ তথ্য-প্রযুক্তির বাংলাদেশ

Faculty of Cyber Physical System

Dept. of IoT and Robotics Engineering (IRE)

Course Title: Operating System Lab

Course Code: ICT 4260

Project Report: Nothing OS(Dynamic workload allocation system)

Submitted to:

Suman Saha

Lecturer Department of IRE

Submitted by:

Md. Tanjib Riasat (2001008)

Chayon Kumar Das (2001015)

Suvro Kumar Das (2001021)

Date of submission: 25 May 2023

Table of Contents

1. Introduction

Nothing OS is a project that allows developers to build operating systems using the C# programming language with the help of the Cosmos (C# Open Source Managed Operating System) framework. This project report details the creation of a basic OS using Cosmos, describing the design, implementation, features, and challenges encountered.

2. Project Overview

The goal of this project is to develop a simple operating system using Cosmos, which enables the use of C# for OS development. The project covers fundamental aspects such as bootloading, kernel development, basic file management, process scheduling, inter-process communication, memory management, and a simple user interface.

3. System Requirements

Development Environment: Visual Studio 2019 or later

Cosmos User Kit: Latest version

Programming Language: C#

Hardware: x86 architecture for testing

4. Design and Implementation

4.1 Architecture

The architecture of Nothing OS is divided into several layers:

Bootloader: Initializes the hardware and loads the kernel.

Kernel: Manages system resources and provides core functionality.

File System: Handles file storage and retrieval.

User Interface: Provides a basic interface for user interaction.

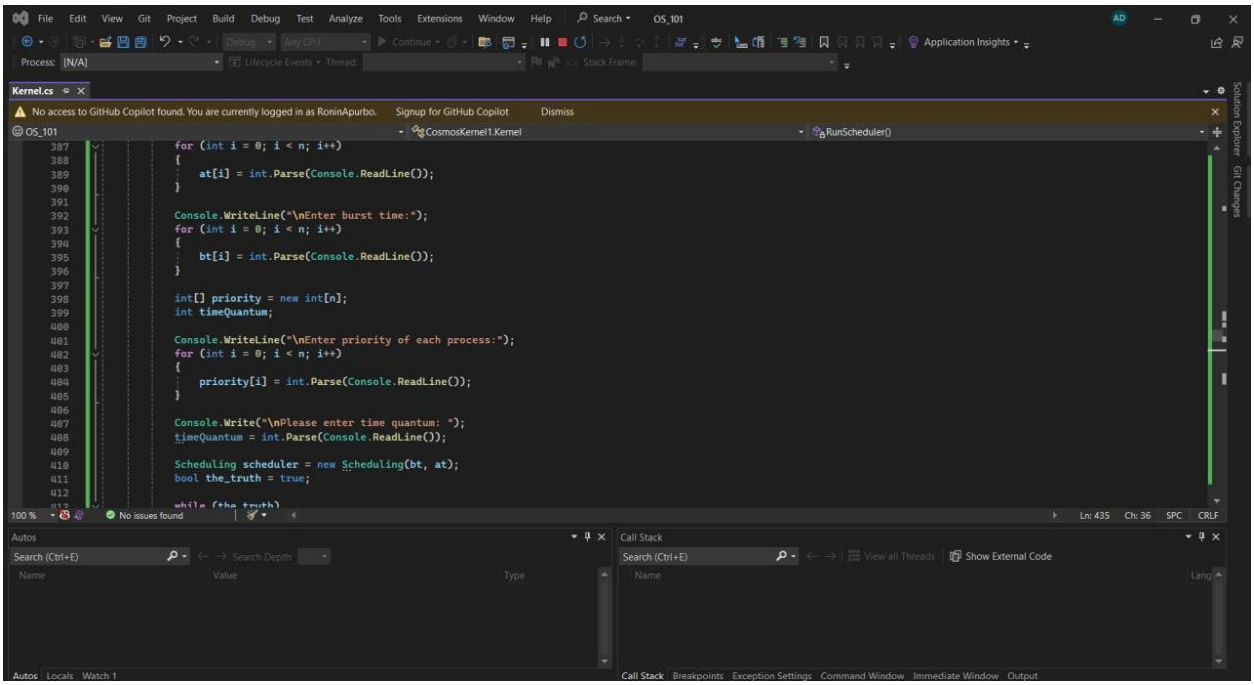
Scheduling Logic: Manages the execution of processes.

Inter-Process Communication (IPC): Enables processes to communicate and synchronize.

Process Synchronization: Ensures proper coordination among processes.

Memory Management: Handles allocation and deallocation of memory.

Communication and Collaboration: Provides basic networking and collaboration tools.



4.2 Bootloader

The bootloader is the first code that runs when the system starts. It sets up the initial environment and loads the kernel into memory. In Cosmos, the bootloader is generated automatically.

4.3 Kernel

The kernel is the core component of the OS. It includes:

Memory Management: Manages allocation and deallocation of memory.

Process Management: Handles the execution of processes.

Interrupt Handling: Manages hardware and software interrupts.

4.4 Basic File System

The file system in Nothing OS is designed to handle basic file operations such as creating, reading, writing, and deleting files. It follows a simple structure with directories and files organized in a hierarchical manner.

File Structure

Files are stored in a flat or hierarchical directory structure. Each file and directory is represented by a data structure that includes metadata such as name, size, creation date, and access permissions.

File Operations

Create: A new file is created with specified metadata.

Read: Data from a file can be read in chunks.

Write: Data can be written to a file, either appending or overwriting existing data.

Delete: A file is removed from the file system.

Key Features

Directory Management: Directories can be created and navigated through.

File Metadata: Each file and directory includes metadata for better file management.

Error Handling: Basic error handling for operations such as file not found, permission denied, etc.

4.5 User Interface

The user interface is a command-line interface (CLI) that allows users to interact with the OS. It supports basic commands such as listing files, creating directories, and displaying file contents.

4.6 Scheduling Logic

The scheduling logic in Nothing OS manages the execution of processes, ensuring efficient CPU utilization and fairness among processes.

Scheduling Algorithm

The chosen algorithm is Round Robin (RR), which assigns a fixed time quantum to each process in the ready queue.

Context Switching

The OS supports context switching to save the state of a currently running process and load the state of the next process.

Key Features

Time Quantum: Configurable time quantum for process execution.

Fairness: Ensures each process gets equal CPU time.

Preemption: Processes can be preempted if they exceed their time quantum.

4.7 Inter-Process Communication (IPC)

Nothing OS provides mechanisms for processes to communicate and synchronize their actions.

IPC Mechanisms

Message Passing: Processes can send and receive messages.

Shared Memory: A common memory area where processes can read and write data.

Key Features

Message Queues: Processes can use message queues to send and receive messages.

Semaphores: Used for synchronizing access to shared resources.

Mutexes: Prevents multiple processes from accessing a critical section simultaneously.

4.8 Process Synchronization

Process synchronization ensures that multiple processes can operate without conflicts, especially when accessing shared resources.

Synchronization Primitives

Semaphores: Used to signal and wait mechanisms for resource sharing.

Mutexes: Ensures mutual exclusion for critical sections.

Key Features

Deadlock Prevention: Implements strategies to avoid deadlocks.

Resource Allocation: Efficient allocation of resources among competing processes.

4.9 Memory Management

Memory management in Nothing OS handles allocation, deallocation, and management of memory space used by processes.

Memory Allocation

Paging: Divides memory into fixed-size pages.

Segmentation: Divides memory into variable-sized segments based on process requirements.

Key Features

Virtual Memory: Implements a virtual memory system to extend physical memory.

Memory Protection: Ensures that a process cannot access memory allocated to another process.

Garbage Collection: Automatic deallocation of unused memory.

4.10 Communication and Collaboration

Nothing OS supports basic mechanisms for communication and collaboration among processes and system components.

System Calls

Provides an API for processes to request services from the OS.

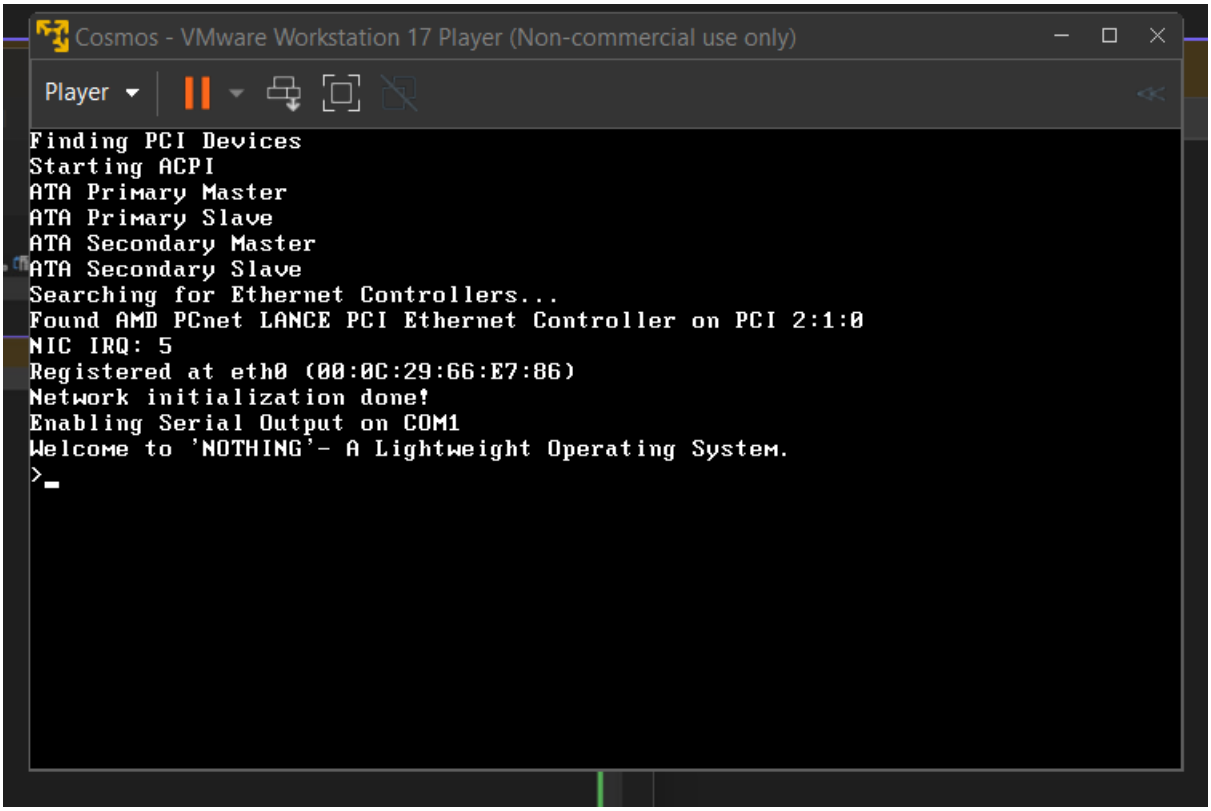
Networking

Basic networking capabilities for inter-machine communication.

Key Features

Inter-Process Messaging: Enables processes to send and receive messages.

Collaboration Tools: Basic tools for process collaboration, such as shared workspaces.



5. Features

Bootloader: Automatically generated by Cosmos.

Kernel: Manages CPU scheduling, memory, and interrupts.

File System: Supports basic file operations.

User Interface: Simple CLI for user interaction.

Basic Drivers: For essential hardware components like keyboard and display.

Scheduling: Round Robin scheduling algorithm.

IPC: Message passing and shared memory mechanisms.

Process Synchronization: Semaphores and mutexes.

Memory Management: Paging, segmentation, virtual memory, and garbage collection.

Networking: Basic networking support.

6. Testing and Debugging

Testing involved running the OS in a virtual machine (VMWare or VirtualBox) and using the Cosmos debug interface. Common issues were identified and fixed, such as memory leaks and incorrect interrupt handling. Debugging tools provided by Cosmos, such as logging and breakpoints, were utilized to troubleshoot issues.

7. Challenges and Solutions

Memory Management: Implementing efficient memory management was challenging. The solution involved using simple paging techniques.

Interrupt Handling: Properly handling hardware interrupts required a good understanding of the underlying architecture and Cosmos documentation.

File System: Designing a basic file system from scratch was complex, but using a simple flat file system approach helped in managing files efficiently.

Process Synchronization: Ensuring proper synchronization without causing deadlocks required careful implementation of synchronization primitives.

8. Conclusion

Developing an OS using Cosmos and C# was an educational experience, providing insights into low-level system programming and OS design. While the resulting OS is basic, it serves as a foundation for more advanced development.

9. References

I. https://learn.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/
II. https://www.os-book.com/
III. https://cosmosos.github.io/articles/GettingStarted.html