Project Report

Faculty of Cyber Physical Systems
Department of IOT and Robotics Engineering

Course Title: Operating Systems
Course Code: ICT 4260

Submitted to:	**Submitted by:**
Suman Saha Lecturer, Dept. of IRE, Bangabandhu Sheikh Mujibur Rahman Digital University,Bangladesh	Shimul Mahmud Khatami ID:2001037 Md Tajbir Ahmed ID:2001046 Md Rakib Hasan ID:1801047

Table of Contents

Topic	Page Number
Overview	2
Objectives	2
Methodology	3
File and storage management	4
Implementing Scheduling Logic	5
Memory Management	8
Basic I/O functions	9
Conclusion	10

OverView

In this project, we are going to build a 64-bit operating system from scratch. Think of it like crafting the brain of a computer! We're using a mix of C# programming to create a simple operating system.

The main goal is to make an efficient OS that can handle basic tasks like managing files, sorting out storage, and deciding which jobs the computer should do first, meaning CPU scheduling. By combining the friendliness of C# with the low-level instructions of assembly language, we're dividing into the nuts and bolts of how computers work. The project is all about understanding and putting into practice the teamwork between hardware and software.

Objectives

1. File Management
2. Storage Management
3. CPU Scheduling
4. Memory management
5. Basic I/O functions

Methodology

1.Kernel Development

- Establish the foundational structure of the kernel using C# programming.
- Implement essential functionalities, such as interrupt handling and system calls, to establish communication between software and hardware.

2.File and Storage Management

- Design and implement a file system for organizing and managing data.
- Developing mechanisms for reading and writing to storage devices.

3.CPU scheduling

- Devise and implement a simple CPU scheduling system algorithm using a combination of C# language.
- Prioritize and execute processes based on the scheduling algorithm to manage system resources effectively.
- Ensure compatibility with diverse hardware configurations.

4.User Interface and Programs

- Create a basic command line interface (CLI) using C for user interaction.

5.Memory Management

- Implement basic memory allocation and deallocation mechanisms to efficiently manage system memory.

6.Testing and Debugging

- Conduct regular testing at each development phase to identify and address bugs and issues promptly.
- Utilize debugging tools and techniques to ensure the stability and reliability of the operating system.

7.Documentation

- Maintain comprehensive documentation throughout the development process, detailing the design choices, algorithms, and code structure.

File and storage management

File management in an operating system (OS) involves various activities related to creating, storing, organizing, retrieving, and managing files and directories.

```
touch
Enter the file name :
lab1.txt
lab1.txt created successfully!

write
Enter the file name :
hello world from lab1
hello world from lab1.txt does not exist. Please create the file first.

write
Enter the file name :
lab1.txt
Enter the text to write to the file:
hello from lab1
Content written to 'lab1.txt' successfully!
```

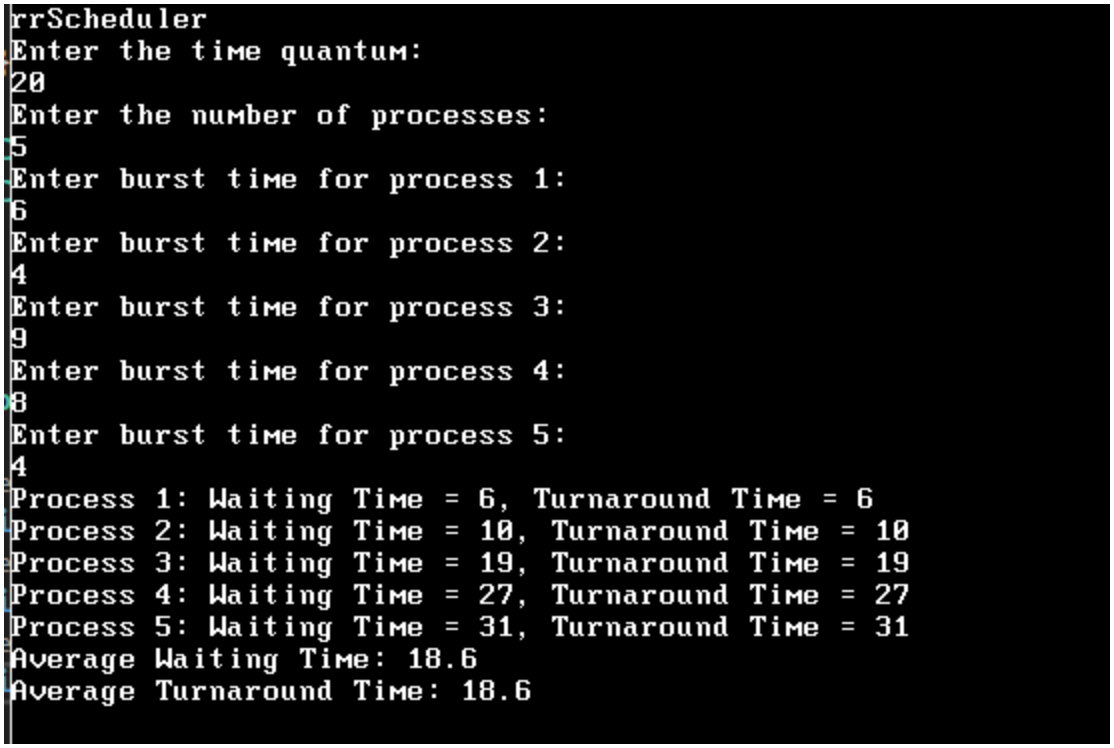
```
show
Enter the file name :
lab1.txt
Content of 'lab1.txt':
hello from lab1

dls
Enter the file name :
lab1.txt
lab1.txt deleted successfully!
```

```
ls
Files in the current directory:
Root.txt
bdu.txt
ananna.txt
hello.txt
cosmos.txt
```

Implementing Scheduling Logic

implementing scheduling logic in an operating system involves designing and implementing algorithms that determine the order in which processes or threads are executed by the CPU. This is crucial for ensuring efficient utilization of CPU resources and providing fair and responsive multitasking.

```
fcfs_scheduler
Enter the number of processes:
3
Enter burst time for process 1:
6
Enter burst time for process 2:
3
Enter burst time for process 3:
8
Process    Burst Time    Waiting Time    Turnaround Time
1    6          0          6
2    3          6          9
3    8          9          17
Average Waiting Time: 5
Average Turnaround Time: 10.666667
```


rrScheduler
Enter the time quantum:
20
Enter the number of processes:
5
Enter burst time for process 1:
6
Enter burst time for process 2:
4
Enter burst time for process 3:
9
Enter burst time for process 4:
8
Enter burst time for process 5:
4
Process 1: Waiting Time = 6, Turnaround Time = 6
Process 2: Waiting Time = 10, Turnaround Time = 10
Process 3: Waiting Time = 19, Turnaround Time = 19
Process 4: Waiting Time = 27, Turnaround Time = 27
Process 5: Waiting Time = 31, Turnaround Time = 31
Average Waiting Time: 18.6
Average Turnaround Time: 18.6

Memory Management

Memory management in an operating system (OS) involves efficiently handling and coordinating computer memory, which includes the main memory (RAM) and, to some extent, secondary storage (like hard drives or SSDs). The primary

goal is to ensure that processes have enough memory to execute while optimizing overall system performance and stability.

Basic I/O functions

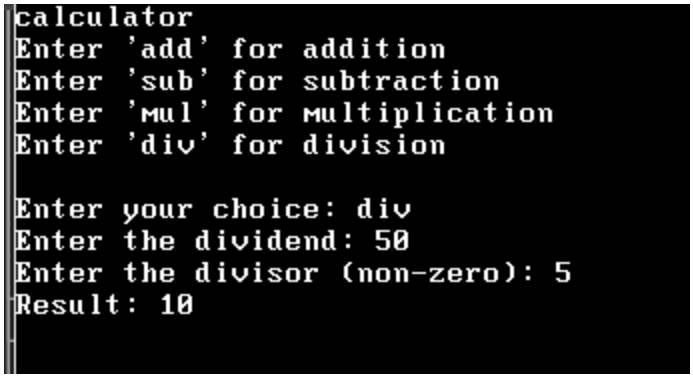
Basic Input/Output (I/O) functions in an operating system (OS) are essential for managing the communication between hardware devices (like keyboards, monitors, disk drives, and network interfaces) and software applications. These

functions ensure that data can be read from and written to various I/O devices in an efficient and standardized manner.

```
calculator
Enter 'add' for addition
Enter 'sub' for subtraction
Enter 'mul' for multiplication
Enter 'div' for division

Enter your choice: add
Enter the first number: 8
Enter the second number: 15
Result: 23
```

```
calculator
Enter 'add' for addition
Enter 'sub' for subtraction
Enter 'mul' for multiplication
Enter 'div' for division

Enter your choice: sub
Enter the first number: 4
Enter the second number: 50
Result: -46
```

```
calculator
Enter 'add' for addition
Enter 'sub' for subtraction
Enter 'mul' for multiplication
Enter 'div' for division

Enter your choice: mul
Enter the first number: 5
Enter the second number: 3
Result: 15
```



Conclusion

This project has not only expanded our technical skills but also deepened our appreciation for the intricacies of operating system design and development.

It has laid a solid foundation for future explorations and innovations in the field, equipping us with the knowledge to tackle even more complex challenges.

In conclusion, the creation of our 64-bit kernel stands as a testament to the power of meticulous design, continuous learning, and the unwavering pursuit of technological advancement. It is a significant step forward in our journey as developers, providing a robust platform for future growth and exploration in operating system development.