IIT Bombay

22M2113

Roll no	22M2113	Name	Sourav Paul
Department	Computer Science and Engineering	Program	Master of Science By Research

Academic Performance Summary

Year	Sem	SPI	CPI	Sem Credits Used for SPI	Completed Semester Credits	Cumulative Credits Used for CPI	Completed Cumulative Credits
2023	Spring	10.0	9.17	6.0	6.0	58.0	58.0
2023	Autumn	9.0	9.08	6.0	6.0	52.0	52.0
2022	Spring	9.67	9.09	18.0	18.0	46.0	46.0
2022	Autumn	8.71	8.71	28.0	28.0	28.0	28.0

Semester-wise Details

*This registration is subject to approval(s) from faculty advisor/Course Instructor/Academic office.

Year/Semester: 2023-24/Spring

Course Code	Course Name	Credits	Tag	Grade	Credit/Audit
CS 765	Introduction to Blockchains, Cryptocurrencies and Smart Contracts	6.0	Department elective	AA	C

Year/Semester: 2023-24/Autumn

Course Code	Course Name	Credits	Tag	Grade	Credit/Audit
CS 742	Foundations of Network Security and Cryptography	6.0	Department elective	AB	C
CS 770	Process Engineering	6.0	Additional Learning	BB	C

Year/Semester: 2022-23/Spring

Course Code	Course Name	Credits	Tag	Grade	Credit/Audit
CS 694	Seminar	4.0	Core course	AA	C

CS 695	Topics in Virtualization and Cloud Computing	6.0	Department elective	AB	C
CS 778	M.S. R&D 2	8.0	Core course	AA	C
CS 899	Communication Skills	6.0	Core course	PP	N

Year/Semester: 2022-23/Autumn

Course Code	Course Name	Credits	Tag	Grade	Credit/Audit
CS 631	Implementation Techniques for Relational Database Systems	6.0	Department elective	AB	C
CS 699	Software Lab.	8.0	Core course	BB	C
CS 725	Foundations of Machine Learning	6.0	Core course	AU	A
CS 744	Design and Engineering of Computing Systems	6.0	Department elective	AB	C
CS 777	M.S. R&D 1	8.0	Core course	AB	C
GC 101	Gender in the workplace	0.0	Core course	PP	N
TA 101	Teaching Assistant Skill Enhancement & Training (TASET)	0.0	Core course	PP	N

Report Problem

Course Project Report:

Implementation of Linux Shell

Course: CS 744: Design and Engineering of Computing Systems
Instructor: Prof. Mythili Vutukuru

Project: Building a Basic UNIX Shell

Objective:

This project aimed to create a foundational UNIX shell capable of executing user commands, managing processes, and interacting with the operating system. The shell was designed to provide a basic command-line interface, mimicking core functionalities of traditional shells while emphasizing process management and user input handling.

Core Functionalities:

- **Command Execution:** The shell supports a range of built-in commands (e.g., `cd`, `echo`, `cat`) and external programs. It parses user input, creates child processes using `fork`, and executes commands using `execvp`.
- **Process Management:** The shell differentiates between foreground and background processes. Foreground tasks block the shell until completion, while background processes run concurrently. The shell employs `wait` to manage child processes and prevent zombie processes.
- **Signal Handling:** The shell gracefully handles `SIGINT` (Ctrl+C) to terminate foreground processes and a custom signal for orderly shutdown.
- **Memory Management:** Efficient memory allocation and deallocation are implemented to prevent memory leaks.
- **Job Control:** Basic job control is supported, allowing the shell to keep track of background processes.

Implementation Details:

- **Input Parsing:** User input is tokenized to extract commands and arguments.
- **Command Execution:** Built-in commands are handled directly within the shell, while external commands are executed in child processes.
- **Process Management:** Forking, exec, and wait system calls are utilized for process creation, execution, and monitoring.
- **Signal Handling:** Custom signal handlers are registered to manage `SIGINT` and the custom exit signal.
- **Memory Management:** Dynamic memory allocation is employed for flexible data handling, with careful attention to deallocation.

Challenges and Learnings:

- **Process Management:** Grasping the intricacies of process creation, termination, and synchronization using system calls was a key learning.
- **Signal Handling:** Understanding signal mechanisms and implementing appropriate handlers to maintain shell behavior was challenging but rewarding.
- **Memory Management:** Effective memory management in C required diligence to prevent resource leaks and ensure program stability.

Overall, this project provided a solid foundation in system programming, process management, and shell design. It deepened understanding of UNIX system calls and their application in building interactive command-line tools