ADVANCED OPERATING SYSTEMS

CSEN383

PROJECT-6 (Pipe Simulator)

Group – 3

-Aman Jain, Amarnath S Kaushik, Daryl Ho, Meghana Kale, Sai Kiran Jasti

1. Introduction

This project demonstrates the use of UNIX I/O system calls to create a multiprocess program that reads from multiple child processes via pipes, handles terminal input, and writes the output to a file. The parent process uses the select() system call to multiplex input from the pipes and terminal.

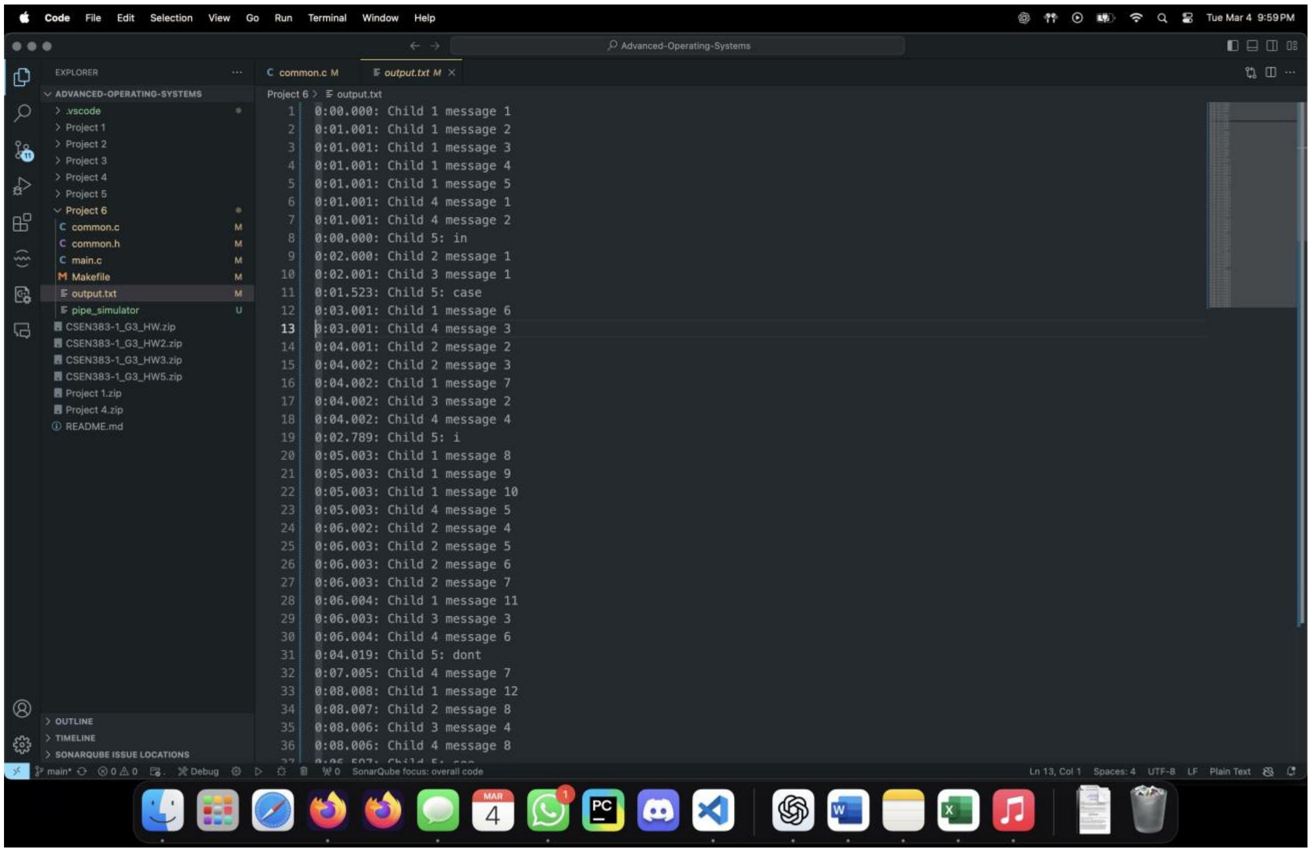
2. Project Overview

The program consists of a parent process and five child processes. Four child processes generate timestamped messages at random intervals, while the fifth child interacts with the terminal. The parent process reads messages from all children and writes them to output.txt.

3. Implementation Details

- **Pipes**: Created using pipe() and used for communication between parent and child processes.
- **Child Processes**: Four children generate messages with timestamps and random sleep intervals. The fifth child reads user input and sends it to the parent.
- **Parent Process**: Uses select() to monitor multiple file descriptors and writes messages to output.txt.
- **Timestamp Formatting**: Timestamps are calculated using gettimeofday() and formatted to the nearest millisecond.

4. Output



```
0:00.000: Child 1 message 1
0:01.001: Child 1 message 2
0:01.001: Child 1 message 3
0:01.001: Child 1 message 4
0:01.001: Child 1 message 5
0:01.001: Child 4 message 1
0:01.001: Child 4 message 2
0:00.000: Child 5: in
0:02.000: Child 2 message 1
0:02.001: Child 3 message 1
0:01.523: Child 5: case
0:03.001: Child 1 message 6
0:03.001: Child 4 message 3
0:04.001: Child 2 message 2
0:04.002: Child 2 message 3
0:04.002: Child 1 message 7
0:04.002: Child 3 message 2
0:04.002: Child 4 message 4
0:02.789: Child 5: i
0:05.003: Child 1 message 8
0:05.003: Child 1 message 9
0:05.003: Child 1 message 10
0:05.003: Child 4 message 5
0:06.002: Child 2 message 4
0:06.003: Child 2 message 5
0:06.003: Child 2 message 6
0:06.003: Child 2 message 7
0:06.004: Child 1 message 11
0:06.003: Child 3 message 3
0:06.004: Child 4 message 6
0:04.019: Child 5: dont
0:07.005: Child 4 message 7
0:08.008: Child 1 message 12
0:08.007: Child 2 message 8
0:08.006: Child 3 message 4
0:08.006: Child 4 message 8
```

5. Conclusion

This project provided valuable experience with UNIX I/O system calls and multiprocess programming. Key takeaways include the importance of proper buffer handling and synchronization in complex systems. Future improvements could include better error handling and support for more child processes.