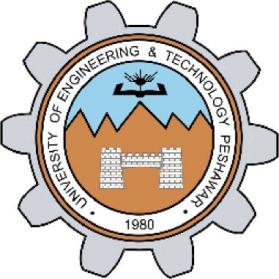
**Project Title:** Operating System Tycoon

Sub Title: Simulation of OS Process Management



Spring 2025

**CSE-204L Operating Systems Lab**

Submitted by: Muhrram Ali

Registration No.: 23PWCSE2279

Muhammad Jalal Khan

Registration No : 23PWCSE2262

Section: A

Submitted to:

**Dr Madiha Sher**

(29th June 2025)

**Department of Computer systems engineering,**

**University of Engineering and Technology, Peshawar**

## ****Objective:****

To design and implement a console-based simulation of an Operating System’s core functions, including process creation, scheduling, memory management, and I/O operations, using the C programming language.

## ****Introduction:****

Modern operating systems are responsible for managing system resources and providing services to user applications. They perform tasks such as process scheduling, memory allocation, and handling I/O operations.  
This project, titled **Operating System Tycoon**, aims to **simulate these core functionalities in a simplified environment**, enabling users to understand and visualize how an OS works internally.

## ****Software and Tools:****

* **Operating System:** Linux (Ubuntu)
* **Programming Language:** C
* **Compiler:** GCC (GNU Compiler Collection)
* **Editor:** nano / vim

## ****Project Features:****

✅ Process creation with user-defined properties  
✅ Round Robin scheduling simulation  
✅ Simulated memory allocation and release  
✅ Simulated I/O wait cycles  
✅ Interactive command-line menu

## ****Implementation Details:****

### **Main Components:**

1. **Process Control Block (PCB):**  
   Each process is represented by a structure containing:
   * Process ID (PID)
   * Name
   * Memory needed
   * CPU time needed
   * CPU time used
   * Current state (READY, RUNNING, WAITING, TERMINATED)
2. **Memory Management:**
   * Simulated as a fixed-size integer variable tracking used memory.
   * Processes cannot start if insufficient memory is available.
3. **Scheduling Algorithm:**
   * A Round Robin scheduler cycles through all READY processes.
   * Each process receives a CPU time slice.
   * After every 2 cycles, the process simulates an I/O operation.
4. **Process States:**
   * **READY:** Waiting for CPU.
   * **RUNNING:** Currently using CPU.
   * **WAITING:** Performing I/O.
   * **TERMINATED:** Completed execution.

### **Workflow:**

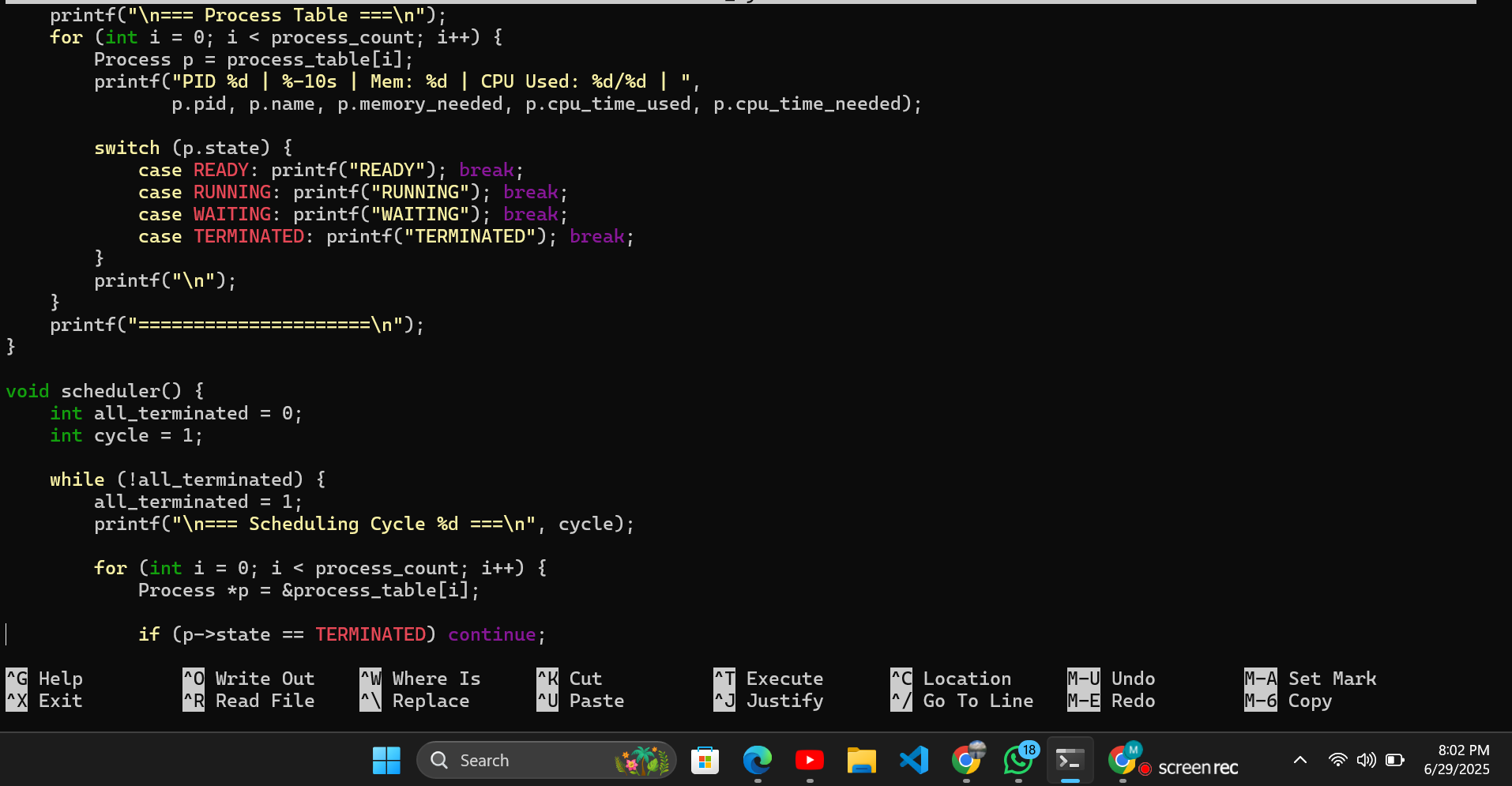
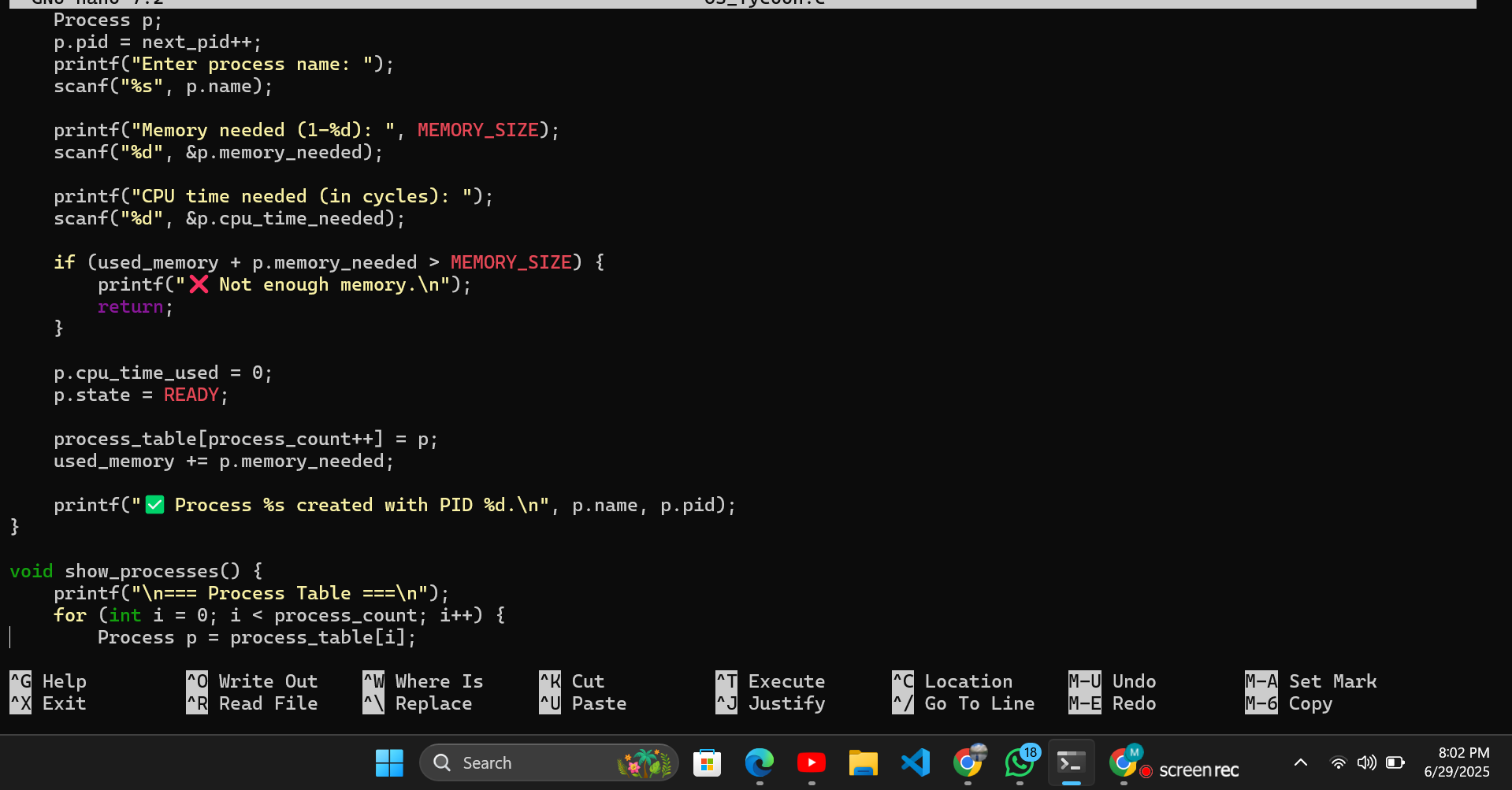
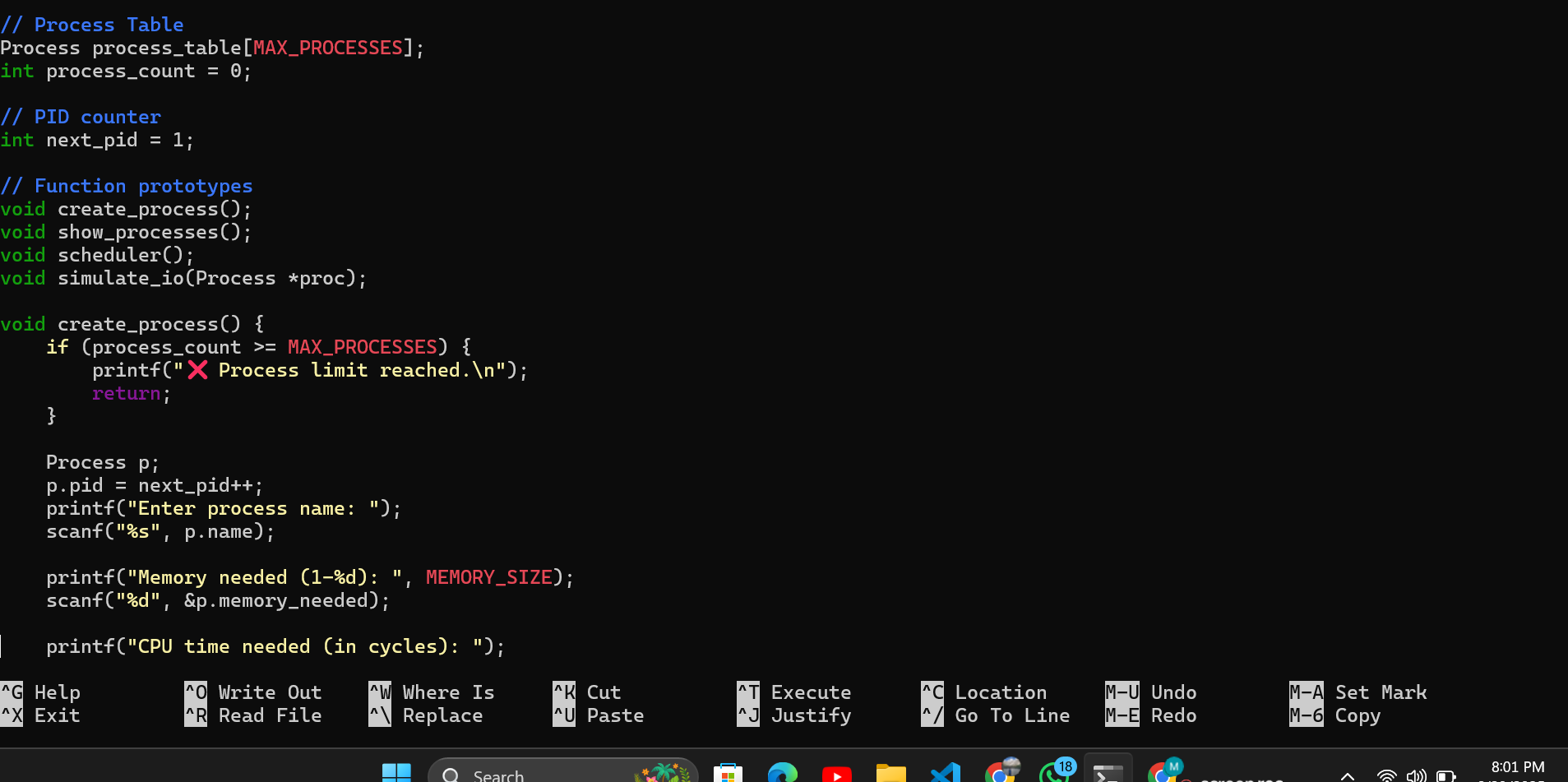
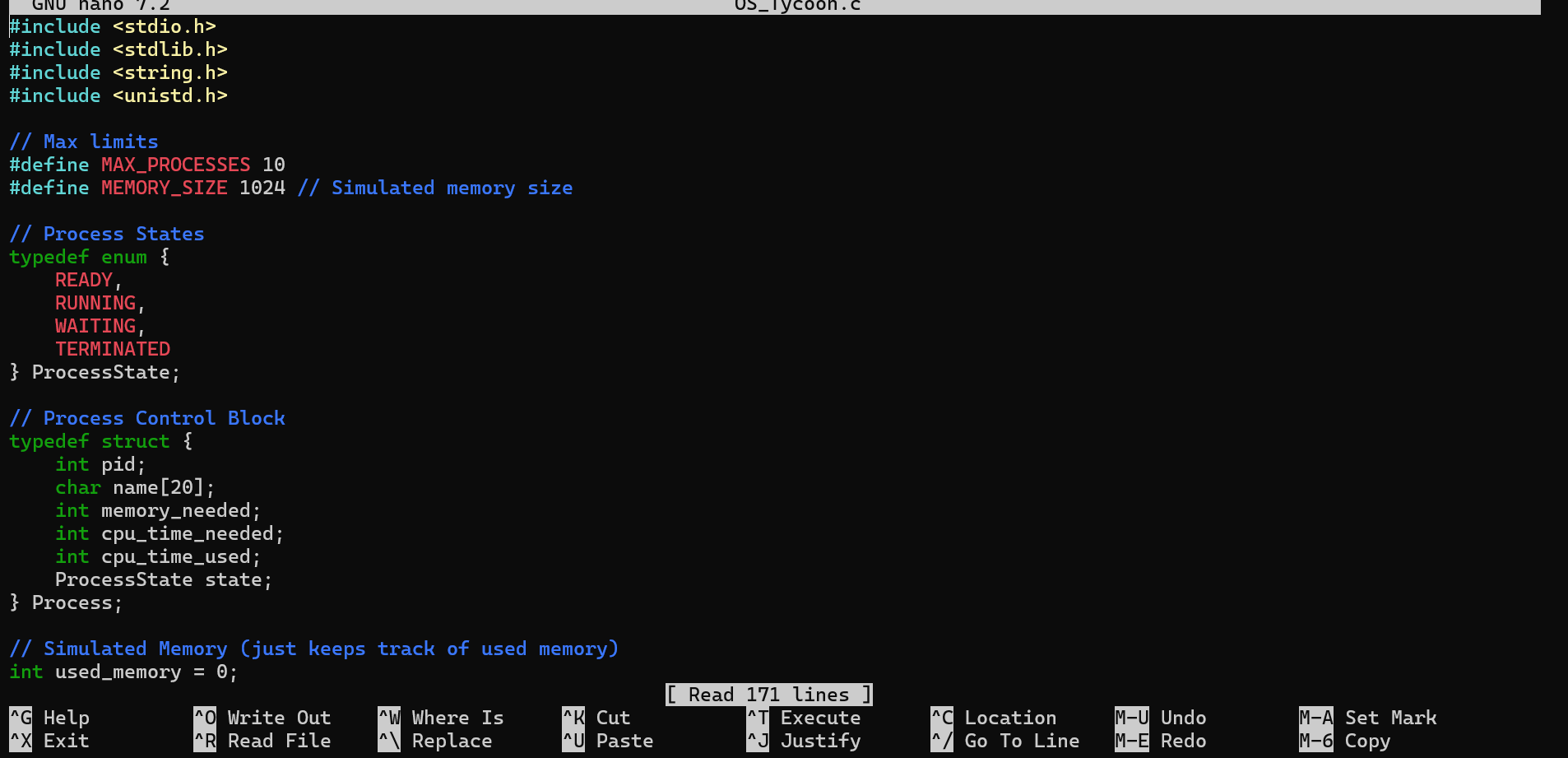
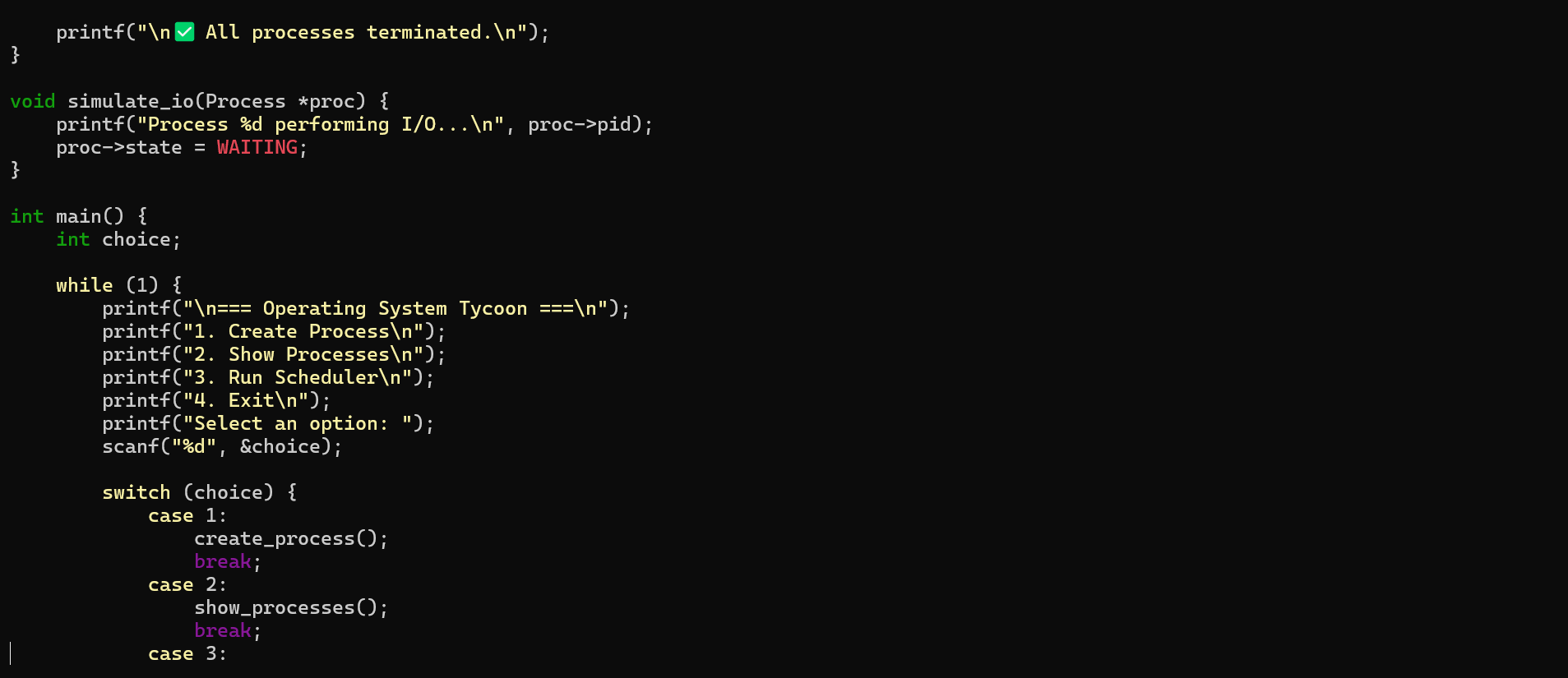
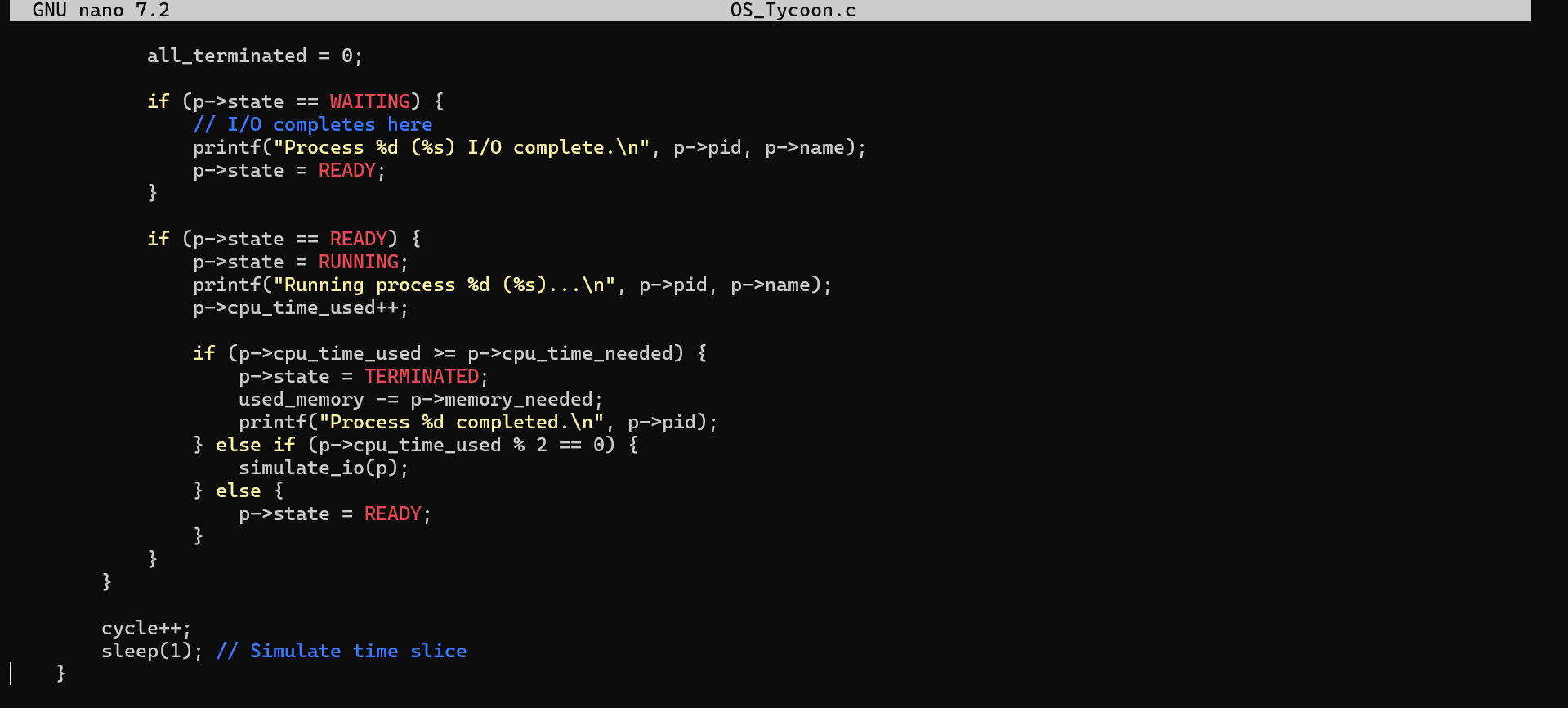
1. **Startup Menu:**
   * Create Process
   * Show Processes
   * Run Scheduler
   * Exit
2. **Process Creation:**
   * User provides name, memory, and CPU time needed.
   * The program checks available memory and adds the process to the table.
3. **Scheduler Simulation:**
   * Iterates over all processes.
   * Assigns CPU cycles.
   * Simulates I/O and termination.
4. **Termination:**
   * When CPU time is used up, the process is marked TERMINATED, and memory is released.

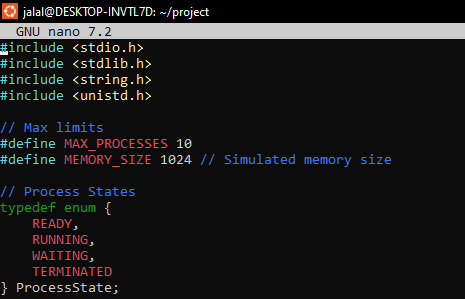
## FLOWCHART:

## UML CHART (unified modeling language):

## 

## Code Snapshot:



## ****Testing and Results:****

**Test Scenario 1:**

* Created 3 processes of varying memory and CPU time.
* Observed correct scheduling, I/O simulation, and termination.

**Test Scenario 2:**

* Tried to create a process exceeding memory limit.
* Program correctly rejected the creation.

**Test Scenario 3:**

* Ran scheduler without any processes.
* Program handled gracefully without errors.

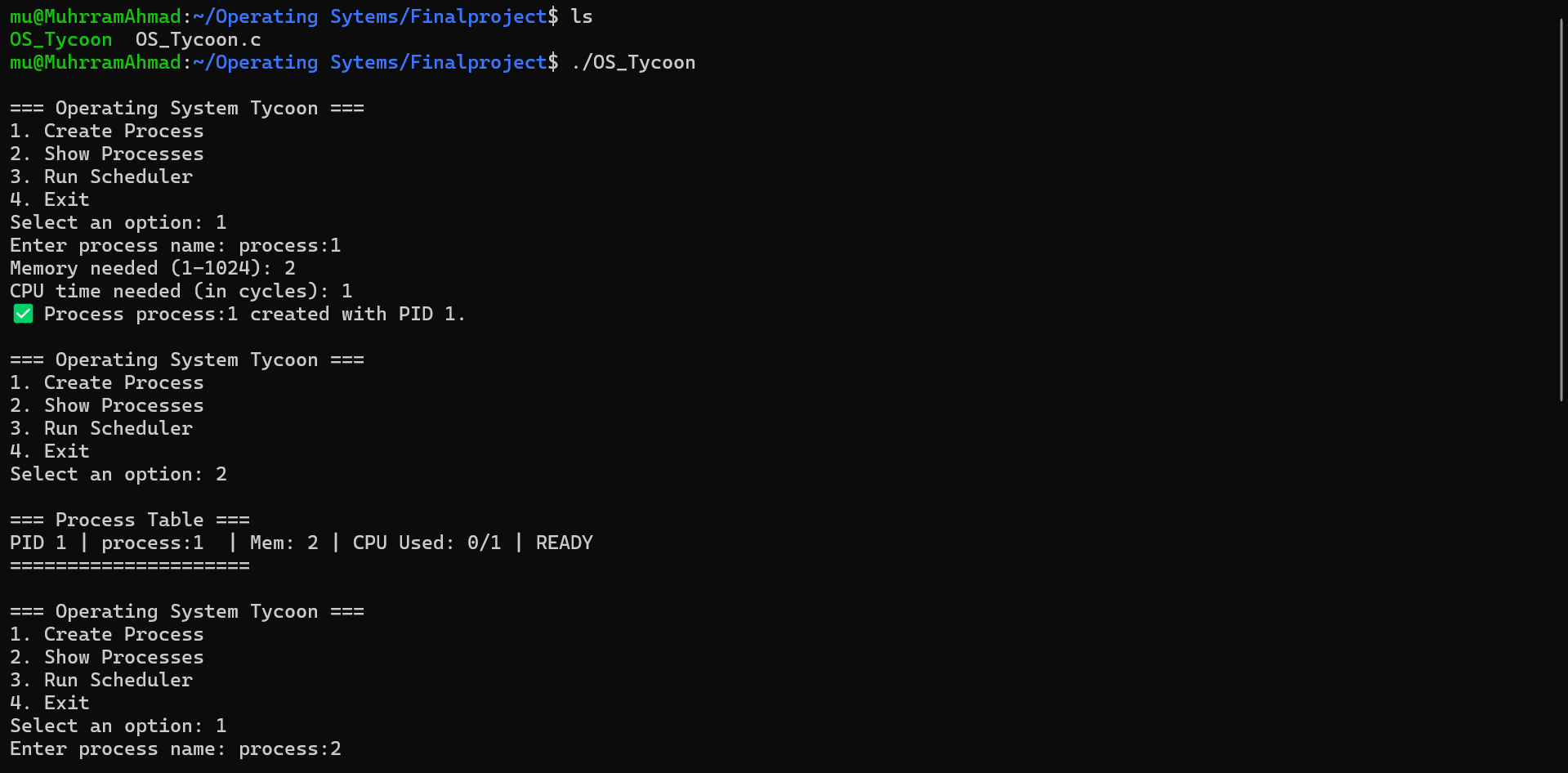
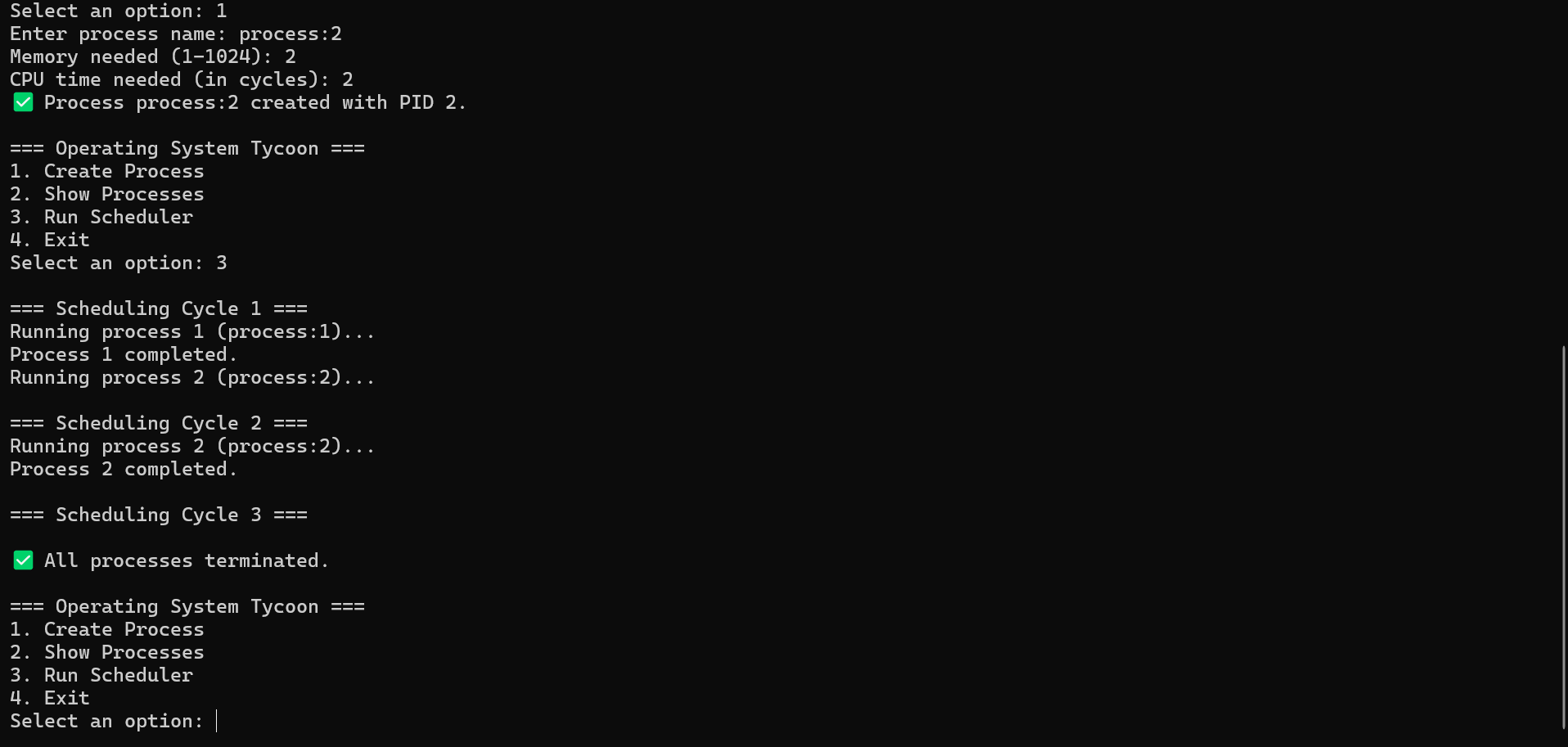
## ****Conclusion:****

This project successfully demonstrates the fundamental aspects of process management in operating systems. Through this simulation, I gained a deeper understanding of:

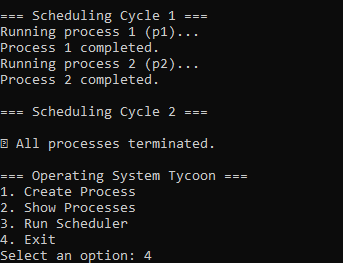
* How processes are created and tracked
* The mechanics of CPU scheduling
* Simulated memory allocation and cleanup
* Handling of I/O operations in a multitasking environment

The project can be extended further by adding:

* Priority-based scheduling
* File management simulation
* Persistent process logs

## 



## ****Appendix:****

**Commands to Compile and Run:**

**bash**

**gcc OS\_Tycoon.c -o OS\_Tycoon**

**./OS\_Tycoon**