006

Threads

Operating Systems

Lab006 Threads

(Windows and Linux)

Objective

The primary objective of this laboratory is to introduce the **concept of threads** and demonstrate **basic thread management** in both **Linux** and **Windows** environments. Through this lab, students will learn how to:

- **Create and manage threads** using **POSIX threads (pthreads)** in **Linux** and **Windows Threads** using **WinAPI**.
- Understand and utilize the **join operation** to synchronize thread execution.
- Observe how threads allow **parallel execution** of tasks, enhancing **application performance**.

Scenario: Multi-Threaded File Processor

You have been tasked with developing a multi-threaded application that processes text files in parallel. The goal is to create a program that counts the number of lines in multiple files simultaneously using separate threads for each file.

The program should:

1. Create a thread for each text file provided as input.
2. Each thread reads a file, counts the lines, and prints the result.
3. The main thread must wait for all threads to complete before displaying a summary of the total line count.

Project Structure:

```
Lab_Threads/
│
├─ Linux/
│   ├─ main_linux.c          # Multi-threaded file processor using POSIX threads
│   ├─ Makefile              # Compilation script for Linux
│   └─ run.sh                # Script to compile and run the Linux program
│
├─ Windows/
│   ├─ main_windows.c        # Multi-threaded file processor using WinAPI
│   └─ run.bat               # Batch script to compile and run the Windows program
│
└─ Instructions/
    └─ Lab_Instructions.pdf  # This lab instruction document
```

Assignment Tasks

Task 1: Create the Multi-Threaded File Processor

1. Create a program that takes a list of file paths as command-line arguments.

Example:

```
./file_processor file1.txt file2.txt file3.txt
```

2. Create a thread for each input file:
    - Linux: Use pthread_create().
    - Windows: Use CreateThread().
3. Each thread should:
    - Open the file in read mode.
    - Count the number of lines in the file.
    - Print the file name and the line count in the format:

      [Thread ID: 1234] file1.txt has 150 lines.
4. The main thread must:
    - Wait for all threads to finish using pthread_join() in Linux or WaitForMultipleObjects() in Windows.
    - Summarize the results by displaying the total line count from all files.

Task 2: Implement Thread Joining

1. Explain the importance of thread joining:
    - Prevents the main program from terminating early.
    - Ensures all files are processed before showing the summary.
2. Add joining logic:
    - In Linux, use:

      ```
      pthread_join(thread_id, NULL);
      ```
    - In Windows, use:

      ```
      WaitForSingleObject(thread_handle, INFINITE);
      ```
3. Test the application with different numbers of files and observe the behavior.

Task 3: Experiment with Delays and Thread Behavior

1. Introduce sleep delays (sleep() in Linux, Sleep() in Windows) to simulate file processing time.
2. Shuffle the file order to see how threads handle concurrent processing.
3. Observe how the order of thread creation may not match the order of execution, demonstrating true concurrency.

Validation Criteria:

1. Correct Output Example:

   [Thread ID: 1234] file1.txt has 100 lines.
   [Thread ID: 5678] file2.txt has 200 lines.
   [Thread ID: 9101] file3.txt has 50 lines.
   Total lines counted: 350

2. The program should handle edge cases, such as:
   - Empty files.
   - Files with only whitespace.
   - Nonexistent files (print an error message).

Expected Learning Outcomes

- Understand the concept of threads and how they differ from processes.
- Gain practical experience in creating, managing, and joining threads.
- Learn how to split tasks across multiple threads to enhance application performance.
- Use thread joining to ensure that the main thread waits for all threads to complete

Additional Resources:

- **Linux POSIX Threads:**
  - pthread_create Documentation
- **Windows Threads:**
  - CreateThread Documentation
- **Multi-Threading Concepts:**
  - Understanding Threads and Concurrency