CSE 344

SYSTEM PROGRAMMING

SPRING 2023

MIDTERM

AHMET CEBECİ

1901042708

INTRODUCTION

The program you provided is a multi-threaded file copying utility written in C using the POSIX threads (pthreads) library. It takes four command line arguments: buffer size, number of threads, source directory, and destination directory. The program copies files from the source directory to the destination directory using a specified number of threads and a buffer of a given size.

SYSTEM ARCHITECTURE

The architecture consists of three main components:

Producer: This is a single thread responsible for scanning the source directory, opening source and destination files, and placing their descriptors into the buffer.

Consumers: These are multiple threads that take file descriptors from the buffer, copy data from the source file to the destination file, and then close the files.

Shared Buffer: This is a buffer that serves as a shared resource between the producer and consumers. It's used to store file descriptors of files to be copied. The buffer has a circular design to efficiently use space.

DESIGN DECISIONS

Several crucial design decisions were made during the implementation of this utility:

Multi-threading: To speed up the file copying process, multiple threads are used to perform the copying in parallel. The number of threads can be customized by the user.

Producer-Consumer Pattern: The producer-consumer design was chosen to efficiently manage the copying of files between directories. This allows the producer to continually scan and add new files to the buffer while the consumers are copying the existing files.

Synchronization: Mutex locks and condition variables are used to ensure safe access to the shared buffer and to handle the synchronization between producer and consumer threads. This is crucial for preventing race conditions and ensuring data consistency.

Recursion: When dealing with directories, the program uses recursion to handle nested directories. This enables the program to copy all files, no matter how deeply they are nested.

IMPLEMENTATION DETAILS

The implementation relies on several POSIX APIs and principles of concurrent programming:

Pthreads: The POSIX threads library is used to create and manage threads.

Mutex and Condition Variables: These are used to synchronize access to the shared buffer. The mutex ensures that only one thread can modify the buffer at a time, while the condition variables allow threads to wait for certain conditions (buffer not empty for consumers, buffer not full for the producer).

File I/O: The utility uses POSIX file I/O functions (open, read, write, close) to handle files.

Shared Buffer: The buffer is implemented as a circular queue to optimize space utilization.

ALGORITHM

The main function first checks the command-line arguments. If the number of arguments is not five (including the program name itself), it prints a usage message and exits.

The buffer size, number of threads, source directory, and destination directory are then read from the command-line arguments.

A "Buffer" is created with the specified buffer size. This Buffer struct includes a circular buffer for storing "BufferItems", along with several other fields for synchronization (mutex, condition variables, etc.) and bookkeeping (in, out, count, done, etc.). A BufferItem struct contains the file descriptors of a source and destination file.

A "ThreadPool" is created with the specified number of threads. This ThreadPool struct includes an array of pthread_t

objects (one for each thread), a count of the total number of threads, and a mutex for synchronization.

The "producer" thread is created and starts running. The producer thread traverses the source directory (recursively, if there are subdirectories), opens each file it finds, and adds the source and destination file descriptors to the Buffer.

The "consumer" threads are also created and start running. Each consumer thread removes a BufferItem from the Buffer, reads from the source file, writes to the destination file, and then closes the file descriptors. If the Buffer is empty, the consumer threads will wait until the producer thread adds a new BufferItem.

When the producer thread finishes adding all the files to the Buffer, it sets the "done" flag in the Buffer to indicate that there are no more files to copy.

Once the "done" flag is set and the Buffer is empty, the consumer threads will stop running.

The main function waits for all threads to finish running (using the pthread_join function), then it prints out some statistics (total files copied, total bytes copied, etc.) and cleans up the resources (Buffer and ThreadPool).

In summary, the producer thread is producing file descriptors to be copied, and the consumer threads are consuming these file descriptors by copying the files. The Buffer is used as a shared resource between the producer and consumers, and access to the Buffer is synchronized using a mutex and condition variables.