CSE-344 HOMEWORK 4

AHMET OKUR

1801042655

Problem Defination

In this project you are expected to implement a directory coping utility “ pCp ”,that creates a new thread to copy each file and sub-directory to perform the overall task in parallel. Similar utilities may quickly exceed system resources when called under a large directory tree. Therefore, in order to regulate the number of active threads at any time, you are expected to implement a worker thread pool. In a worker thread pool implementation, a fixed number of threads are made available to handle the load. The workers block on a synchronization point (in our case, maybe an empty buffer)and one worker unblocks when a request arrives (an item is put in the buffer). The overall implementation of the utility should use a producer-consumer based synchronization.

Plan & Design

This C program copies the contents of a directory to another directory using multi-threading. The code manages concurrency using a producer-consumer model. It also checks and reports error conditions.

Producer

```
for(;;)
    lock(m)
    while(count == N)
        cwait(empty,m)
    produce_item()
    count++
    broadcast(full)
    unlock(m)
```

Consumer

```
    for(;;)
        lock(m)
        while(count == 0)
            cwait(full, m)
        consume_item
        count--
        broadcast(empty)
        unlock(m)
```

- **main():** It is the main entry point of the program. This function first checks the arguments received from the command line. These arguments include the source and destination file paths, the size of the thread pool, and the buffer size. Sets a handler to be called later when the SIGINT signal (Ctrl+C) arrives. This is used to properly clear the memory in case the user stops the program. It then creates a thread pool of the specified size and shares jobs between those threads. It also calculates how long the program takes to complete and reports how many files and directories have been copied.
- **consumer():** This function is run by each consumer thread. As long as there are elements in the queue, the thread takes a file from the queue and copies it to the target location. The thread terminates when the queue is empty or when the program terminates.
- **producer():** This function is executed by the producer thread. The function scans every file and subdirectory within the specified source directory and creates a copy task for each file. Each task is performed by consumer threads.

```
pthread_mutex_lock(&mutex);
while (count == N)
{
    pthread_cond_wait(&empty, &mutex);
}

int sourceFile = open(sourcePath, O_RDONLY);
int destFile = open(destPath, O_WRONLY | O_CREAT, 0644);
enqueue(queue, sourceFile, destFile);
count++;

pthread_cond_broadcast(&full);
pthread_mutex_unlock(&mutex);
```

(This part is inside the copy Directory method)

- **copyFile():** This function copies a source file to the specified destination location. This is accomplished by reading a certain amount of data from the source file and writing that data to the destination file.

- **copyDirectory():** This function copies a source directory and all its subdirectories and files to the specified destination location. This process involves creating the same structure in the target location and copying each file to the appropriate location.
- **handler():** This handler is called when a SIGINT signal (Ctrl+C) arrives. Handler includes stopping threads and freeing resources.

```
struct sigaction sa;    incomplete t
memset(&sa, 0, sizeof(sa));    incom
sa.sa_handler = &handler;    incompl
sigaction(SIGINT, &sa, NULL);
```

(main)

```
void handler(int signal_number)
{
    write(1, "Signal handled\n", 16);
    stopThread = 0;
    free(queue);
}
```

(handler function)

How to run the program

```
raskolnikov@raskolnikov:~/Desktop/hw5sys$ make
gcc pCp.c -o pCp queue.c
raskolnikov@raskolnikov:~/Desktop/hw5sys$ ./pCp 3 3 /home/raskolnikov/Desktop/hw5sys /home/raskolnikov/Desktop/system_ödevler
```

./pCp bufferSize poolSize sourceFile destFile