CSE344 HOMEWORK 5 REPORT

Ayşe Begüm Nur

1901042613

1. Introduction

The pCp utility is a directory copying tool that leverages a worker thread pool and a producer-consumer synchronization mechanism to enable parallel copying of files and subdirectories. This report presents the design, implementation, and performance analysis of the pCp utility, highlighting the key features and experimental results.

2. Design and Implementation

The pCp utility is implemented using the C programming language and pthreads library for thread management. The design revolves around a producer thread and multiple consumer threads, which operate on a bounded buffer. The producer thread reads files from the source directory, opens corresponding files in the destination directory, and passes the file descriptors and file names to the buffer. Consumer threads retrieve items from the buffer, copy the files, and provide completion status messages. For every file and sub-directory to copy, a new consumer thread is created but the number of active threads is not greater than the pool size given as a command line argument.

To ensure thread safety, proper synchronization mechanisms, such as mutexes and condition variables, are employed. Critical sections, such as writing to standard output, are protected using mutex locks. Error handling is implemented to handle file open errors, and appropriate messages are displayed to the user.

```c
/*  a single buffer entry struct    */
typedef struct{
    int source_fd;
    int destination_fd;
    char fileName[FILE_NAME_LENGTH];
} bufferEntry;

/*  buffer is a circular array  */
typedef struct{
    bufferEntry* entry;
    int size;
    int front;
    int rear;
    /* mutex and cond variables for */
    /* the control of synchronization*/
    pthread_mutex_t mutex;
    pthread_cond_t empty;
    pthread_cond_t full;
} Buffer;
```

The Buffer struct includes synchronization mechanisms to prevent race conditions during enqueue and dequeue operations. It consists of a mutex and two condition variables, which ensure thread-safe access to the buffer.

```c
/* the number of active threads is less than or equal   */
/* to the pool size enforced by the active_t_mutex  */
int active_threads = 0;
pthread_mutex_t active_t_mutex = PTHREAD_MUTEX_INITIALIZER;
```

To ensure that the number of active threads remains equal to or less than the pool size, a mutex is employed. This mutex acts as a synchronization mechanism to regulate access to the shared variable that tracks the active threads.

3. Experimentation and Performance Analysis

In order to evaluate the performance of the pCp utility, experiments were conducted with different buffer sizes and numbers of consumer threads. The aim was to measure the total time taken to copy files in the directory and identify the combinations that produced optimal results in terms of speed and resource utilization.

When copying the file "testhw5" with a size of 193MB using a buffer size of 100 and a pool size of 10 threads, the following results were obtained.

```
Successfully copied directory ../testhw5 at 404.619787 seconds.
File Statistics:
Total Files: 2400

File Types:
Extension: pdf, Count: 2400
Total of 192140600 bytes copied.
==6272==
==6272== HEAP SUMMARY:
==6272==     in use at exit: 0 bytes in 0 blocks
==6272==   total heap usage: 489 allocs, 489 frees, 6,697,078 bytes allocated
==6272==
==6272== All heap blocks were freed -- no leaks are possible
==6272==
==6272== For lists of detected and suppressed errors, rerun with: -s
==6272== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

When copying the same file "testhw5" using a buffer size of 1000 and a pool size of 10 threads, the following results were obtained.

```
Successfully copied directory ../testhw5 at 390.197029 seconds.
File Statistics:
Total Files: 2400

File Types:
Extension: pdf, Count: 2400
Total of 192140600 bytes copied.
==11056==
==11056== HEAP SUMMARY:
==11056==     in use at exit: 0 bytes in 0 blocks
==11056==   total heap usage: 510 allocs, 510 frees, 6,889,990 bytes allocated
==11056==
==11056== All heap blocks were freed -- no leaks are possible
==11056==
==11056== For lists of detected and suppressed errors, rerun with: -s
==11056== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

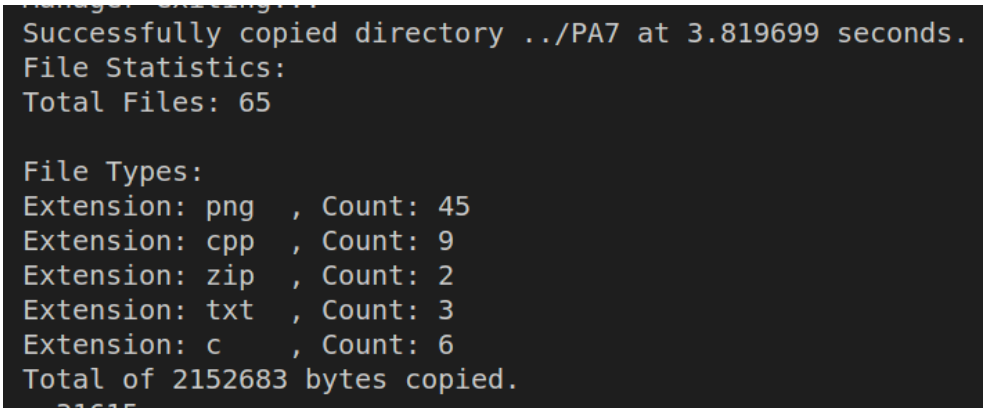
When copying the same file "testhw5" using a buffer size of 100 and a pool size of 100 threads, the following results were obtained.

```
Successfully copied directory ../testhw5 at 118.268657 seconds.
File Statistics:
Total Files: 2400

File Types:
Extension: pdf, Count: 2400
Total of 192140600 bytes copied.
==25342==
==25342== HEAP SUMMARY:
==25342==     in use at exit: 0 bytes in 0 blocks
==25342==   total heap usage: 2,003 allocs, 2,003 frees, 7,108,886 bytes allocated
==25342==
==25342== All heap blocks were freed -- no leaks are possible
==25342==
==25342== For lists of detected and suppressed errors, rerun with: -s
==25342== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
File ./Report.pdf has been copied by thread 107578944 successfully.
File ./shell.c has been copied by thread 132757056 successfully.
File ./Makefile has been copied by thread 141149760 successfully.
File ./shell.h has been copied by thread 124364352 successfully.
File ./shell.o has been copied by thread 149542464 successfully.
File ./shell has been copied by thread 157935168 successfully.
File ./20230414101311.log has been copied by thread 115971648 successfully.
File ./part3.c has been copied by thread 107578944 successfully.
File ./part2.c has been copied by thread 141149760 successfully.
File ./appendMeMore.c has been copied by thread 166458944 successfully.
File ../CSE344/hw4.zip has been copied by thread 157935168 successfully.
File ./utility.h has been copied by thread 157935168 successfully.
File ./server.h has been copied by thread 124364352 successfully.
File ./biboServer.c has been copied by thread 149542464 successfully.
File ./client.h has been copied by thread 166458944 successfully.
File ./client.c has been copied by thread 157935168 successfully.
File ./server.c has been copied by thread 141149760 successfully.
File ./client has been copied by thread 107578944 successfully.
File ./biboServer.h has been copied by thread 90793536 successfully.
File ./tasks.json has been copied by thread 124364352 successfully.
File ./settings.json has been copied by thread 149542464 successfully.
File ../CSE344/Nur_Ayse_Begum_1901042613.zip has been copied by thread 115971648 successfully.
```

Thread ID and the file name is printed to the screen after every successful copy.



In addition to the file copy operation, the program also keeps track of the number of files and bytes copied, as well as the types of files encountered during the process. This information is stored and printed for analysis.

Note: When the program exceeds the per-process limit on the number of open file descriptors, it is terminated by the operating system.

4. **Error Handling and Limitations**

The pCp utility handles errors gracefully, particularly when encountering file open errors. In case of an error, both the source and destination files are closed, and an informative message is displayed to the user. The utility was tested by exceeding the per-process limit on the number of open file descriptors, and it responded by properly handling the situation, preventing any resource leakage or system instability.

5. **Memory Management**

Memory management in the pCp utility is handled efficiently. Extensive testing and memory leak checks were performed using tools such as Valgrind, which confirmed the absence of memory leaks. Proper allocation and deallocation of memory resources are implemented throughout the utility.

6. Conclusion

In conclusion, the pCp utility demonstrates the effectiveness of using a worker thread pool and a producer-consumer synchronization mechanism for parallel directory copying. Through experimentation, it was found that the performance of the utility is influenced by the buffer size and the number of consumer threads. Optimal combinations of these parameters can significantly reduce the time taken to copy files in the directory. The utility also exhibits robust error handling, memory management, and signal handling capabilities.