CSE 344 System Programming HW4 REPORT

Student Name: AYKUT SERT

Student ID: 200104004104

General Structure and Functioning

Our program uses multithread to copy files from the specified source directory to the target directory. The program creates a certain number of worker threads and coordinates file copy operations using a buffer. The program also terminates safely by capturing the Ctrl+C signal during the copying process and notifies the user of the intermediate results.

Pseudocode

Main Function

1 -> Check the arguments and make sure they are valid.

2 -> Get buffer size and number of worker threads.

3 -> Initialize buffer and synchronization structures.

4 -> Set the signal handler to capture the Ctrl+C signal.

5 -> Save start time.

6 -> Create the thread that will start the manager.

7 -> Create worker threads.

8 -> Wait for the manager to complete.

9 -> Wait for worker threads to complete.

10 -> Save end time.

11 -> Perform cleanups and print statistics.

Manager Function

1 -> Get source and target directories.

2 -> Scan the source directory and update the buffer for each file.

3 -> Set the done flag and wake up worker threads.

Worker Function

While:

1 -> Check if the buffer is empty.

2 -> Exit if the buffer is empty and the done flag is set.

3 -> Get file information from buffer.

4 -> Copy the files.

5 -> Update the number of files and bytes copied.

Buffer Functions

initialize_buffer(size): Start buffer.

buffer_is_empty(): Check if the buffer is empty.

buffer_is_full(): Check if the buffer is full.

buffer_add(src_fd, dst_fd, src_name, dst_name): Add file information to buffer.

buffer_get(): Get file information from buffer.

Directory Traversal Function

1 -> Open the source directory.

2 -> Create the target directory.

Scan directory contents:

1 -> If it's a subdirectory, process it recursively.

2 -> If it is a file, add the file information to the buffer.

3 -> If FIFO file, update counters.

Copy Function

1 -> Copy data from source file to target file using buffer.

2 -> Update the number of bytes copied.

Signal Handler

When Ctrl+C signal is received:

1 -> Set the done flag.

2 -> Wake up all threads.

3 -> Wait for worker threads to complete.

4 -> Perform cleanups and print statistics.

5 -> End the program.

Error Handling

handle_error(message): Print the error message and terminate the program.

Some Test Outputs:

Test1: valgrind ./MWCp 10 10 ../testdir/src/libvterm ../tocopy, #memory leak checking, buffer size=10, number of workers=10, sourceFile, destinationFile

```
==13101== Memcheck, a memory error detector
==13101== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==13101== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info
==13101== Command: ./MWCp 10 10 ../testdir/src/libvterm ../tocopy/
==13101==
----------------STATISTICS--------------------
Consumers: 10 - Buffer Size: 10
Number of Regular Files: 194
Number of Directories: 7
Number of FIFO Files: 0
TOTAL BYTES COPIED: 25009680
TOTAL TIME: 0.606 seconds
==13101==
==13101== HEAP SUMMARY:
==13101==     in use at exit: 0 bytes in 0 blocks
==13101==   total heap usage: 22 allocs, 22 frees, 348,624 bytes allocated
==13101==
==13101== All heap blocks were freed -- no leaks are possible
==13101==
==13101== For lists of detected and suppressed errors, rerun with: -s
==13101== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Test2: ./MWCp 10 4 ../testdir/src/libvterm/src ../toCopy #buffer size=10, number of workers=4, sourceFile, destinationFile

```
----------------STATISTICS--------------------
Consumers: 4 - Buffer Size: 10
Number of Regular Files: 140
Number of Directories: 2
Number of FIFO Files: 0
TOTAL BYTES COPIED: 24873082
TOTAL TIME: 0.020 seconds
```

Test3: ./MWCp 10 10 ../testdir ../toCopy #buffer size=10, number of workers=10, sourceFile, destinationFile

```
----------------STATISTICS--------------------
Consumers: 10 - Buffer Size: 10
Number of Regular Files: 3116
Number of Directories: 151
Number of FIFO Files: 0
TOTAL BYTES COPIED: 73520554
TOTAL TIME: 0.094 seconds
```