

REPORT:

Same as HW4 but barriers are added.

```
pthread_barrier_t barrier;    // Barrier to synchronize threads
```

In main : num_workers + manager(1) size barrier.

```
// Initialize buffer and barrier
init_buffer(&buffer, buffer_size);
pthread_barrier_init(&barrier, NULL, num_workers + 1);
```

```
// Manager thread function
void *manager_thread(void *arg) {
    char *source_dir = ((char **)arg)[0];
    char *dest_dir = ((char **)arg)[1];

    gettimeofday(&start_time, NULL); // Record start time
    //printf("Manager thread (Thread ID: %ld) started.\n", syscall(SYS_gettid));
    process_directory(source_dir, dest_dir); // Start processing directories

    pthread_mutex_lock(&buffer.mutex);
    buffer.done = 1;
    pthread_cond_broadcast(&buffer.not_empty);
    pthread_mutex_unlock(&buffer.mutex);
    //printf("Manager thread (Thread ID: %ld) signaling completion.\n", syscall(SYS_gettid));

    ✨ pthread_barrier_wait(&barrier); // Wait for all threads to reach the barrier

    return NULL;
}
```

After file copying, barrier_wait is there.

```
//printf("Copying file: %s to %s (Thread ID: %ld)\n", file.source_path, file.dest_path, syscall(SYS_gettid));
char buf[1024];
ssize_t n;
while ((n = read(source_fd, buf, sizeof(buf))) > 0) {
    if (write(dest_fd, buf, n) != n) {
        perror("Error writing to file");
        break;
    }
    pthread_mutex_lock(&total_bytes_mutex); // Lock mutex for updating total bytes copied
    total_bytes_copied += n;
    pthread_mutex_unlock(&total_bytes_mutex); // Unlock mutex
}
close(source_fd);
close(dest_fd);
}

pthread_barrier_wait(&barrier);
//printf("Worker thread (Thread ID: %ld) exiting after barrier.\n", syscall(SYS_gettid));

return NULL;
}
```

In the end of main barrier is freed.

```
// Clean up resources
free(workers);
destroy_buffer(&buffer);
pthread_barrier_destroy(&barrier);
```

Signal handler (CTRL + C):

```
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/hw5/put_your_codes_
here$ make test3
Running Test 3: buffer size=10, number of workers=100
./MWCp 10 100 ../testdir ../toCopy
^CSignal 2 received, cleaning up...

-----STATISTICS-----
Consumers: 100 - Buffer Size: 10
Number of Regular Files: 784
Number of Directories: 20
Number of FIFO Files: 0
TOTAL BYTES COPIED: 11376698
TOTAL TIME: 1.895 seconds
```

TEST1:

No memory leak.

```

put_your_codes_here > C MWCP.c > main(int, char**)
211 int main(int argc, char *argv[]) {
212     pthread_barrier_t barrier;
213     pthread_barrier_init(&barrier, NULL, num_workers + 1);
214
215     // Create manager thread
216     pthread_create(&manager, NULL, manager_thread, (void *) (char *) {source_dir, dest_dir});
217     for (int i = 0; i < num_workers; i++) {
218         pthread_create(&workers[i], NULL, worker_thread, NULL);
219     }
220
221     // Wait for manager and worker threads to complete
222     pthread_join(manager, NULL);
223     for (int i = 0; i < num_workers; i++) {
224         pthread_join(workers[i], NULL);
225     }
226
227     // Clean up resources
228     free(workers);
229     destroy_buffer(&buffer);
230     pthread_barrier_destroy(&barrier);
231
232     gettimeofday(&end_time, NULL); // Record end time
233     long seconds = end_time.tv_sec - start_time.tv_sec;
234     long microseconds = end_time.tv_usec - start_time.tv_usec;
235     double elapsed = seconds + microseconds * 1e-6;
236 }

```

```

(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/hw5/put_your_codes_here$ make test1
Running Test 1 with Valgrind: buffer size=10, number of workers=10
valgrind --leak-check=full ./MWCP 10 10 ../testdir/src/libvterm ../tocopy
==1263== Memcheck, a memory error detector
==1263== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==1263== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==1263== Command: ./MWCP 10 10 ../testdir/src/libvterm ../tocopy
==1263==

-----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: 4.742 seconds
==1263==
==1263== HEAP SUMMARY:
==1263==    in use at exit: 0 bytes in 0 blocks
==1263== total heap usage: 22 allocs, 22 frees, 348,544 bytes allocated
==1263==
==1263== All heap blocks were freed -- no leaks are possible
==1263==
==1263== For lists of detected and suppressed errors, rerun with: -s
==1263== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/hw5/put_your_codes_here$

```

TEST 2:

```

put_your_codes_here > C MWCP.c > worker_thread(void *)
150 void *worker_thread(void *arg) {
151     perror("Error writing to file");
152     break;
153 }
154
155 pthread_mutex_lock(&total_bytes_mutex); // Lock mutex for updating total bytes copied
total_bytes_copied += n;
pthread_mutex_unlock(&total_bytes_mutex); // Unlock mutex
156
157 close(source_fd);
158 close(dest_fd);
159 }
160
161 pthread_barrier_wait(&barrier);
162 //printf("worker thread (Thread ID: %ld) exiting after barrier.\n", syscall(SYS_gettid));
163
164 return NULL;
165 }
166
167 int main(int argc, char *argv[]) {
168     if (argc != 5) {
169         fprintf(stderr, "Usage: %s <buffer size> <number of workers> <source dir> <dest dir>\n");
170         exit(EXIT_FAILURE);
171     }
172
173     struct sigaction sa;
174     memset(&sa, 0, sizeof(sa));
175     sa.sa_handler = signal_handler;
176     sigaction(SIGINT, &sa, NULL);
177
178     int buffer_size = atoi(argv[1]);
179     num_workers = atoi(argv[2]);
180     char *source_dir = argv[3];
181     char *dest_dir = argv[4];
182
183     pthread_t manager;
184     pthread_t *workers = malloc(num_workers * sizeof(pthread_t));
185
186     init_buffer(&buffer, buffer_size);
187     pthread_barrier_init(&barrier, NULL, num_workers + 1);
188 }

```

```

(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/hw5/put_your_codes_here$ make
make: Nothing to be done for 'all'.
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/hw5/put_your_codes_here$ make test2
Running Test 2: buffer size=10, number of workers=4
./MWCP 10 4 ../testdir/src/libvterm/src ../tocopy
-----STATISTICS-----
Consumers: 4 - Buffer Size: 10
Number of Regular Files: 140
Number of Directories: 2
Number of FIFO Files: 0
TOTAL BYTES COPIED: 24873882
TOTAL TIME: 4.534 seconds
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/hw5/put_your_codes_here$

```

TEST3:

```

put_your_codes_here > C MWCP.c > worker_thread(void *)
150 void *worker_thread(void *arg) {
151     perror("Error writing to file");
152     break;
153 }
154
155 pthread_mutex_lock(&total_bytes_mutex); // Lock mutex for updating total bytes copied
total_bytes_copied += n;
pthread_mutex_unlock(&total_bytes_mutex); // Unlock mutex
156
157 close(source_fd);
158 close(dest_fd);
159 }
160
161 pthread_barrier_wait(&barrier);
162 //printf("worker thread (Thread ID: %ld) exiting after barrier.\n", syscall(SYS_gettid));
163
164 return NULL;
165 }
166
167 int main(int argc, char *argv[]) {
168     if (argc != 5) {
169         fprintf(stderr, "Usage: %s <buffer size> <number of workers> <source dir> <dest dir>\n");
170         exit(EXIT_FAILURE);
171     }
172
173     struct sigaction sa;
174     memset(&sa, 0, sizeof(sa));
175     sa.sa_handler = signal_handler;
176     sigaction(SIGINT, &sa, NULL);
177
178     int buffer_size = atoi(argv[1]);
179     num_workers = atoi(argv[2]);
180     char *source_dir = argv[3];
181     char *dest_dir = argv[4];
182
183     pthread_t manager;
184     pthread_t *workers = malloc(num_workers * sizeof(pthread_t));
185
186     init_buffer(&buffer, buffer_size);
187     pthread_barrier_init(&barrier, NULL, num_workers + 1);
188 }

```

```

(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/hw5/put_your_codes_here$ make test3
Running Test 3: buffer size=10, number of workers=100
./MWCP 10 100 ../testdir ../tocopy
-----STATISTICS-----
Consumers: 100 - Buffer Size: 10
Number of Regular Files: 3116
Number of Directories: 151
Number of FIFO Files: 0
TOTAL BYTES COPIED: 73528554
TOTAL TIME: 13.846 seconds
(base) erent@DESKTOP-E56IRIT:/mnt/c/Users/erent/Desktop/hw5/put_your_codes_here$

```

Barrier working well as we can see. If you uncomment printf's you can get output like this.

```
Worker thread (Thread ID: 2269) exiting after barrier.
Worker thread (Thread ID: 2318) exiting after barrier.
Worker thread (Thread ID: 2257) exiting after barrier.
Worker thread (Thread ID: 2294) exiting after barrier.
Worker thread (Thread ID: 2300) exiting after barrier.
Worker thread (Thread ID: 2302) exiting after barrier.
Worker thread (Thread ID: 2291) exiting after barrier.
Worker thread (Thread ID: 2285) exiting after barrier.
Worker thread (Thread ID: 2289) exiting after barrier.
Worker thread (Thread ID: 2278) exiting after barrier.
Worker thread (Thread ID: 2288) exiting after barrier.
Worker thread (Thread ID: 2301) exiting after barrier.
Worker thread (Thread ID: 2342) exiting after barrier.
Worker thread (Thread ID: 2280) exiting after barrier.
Worker thread (Thread ID: 2305) exiting after barrier.
Worker thread (Thread ID: 2316) exiting after barrier.
Worker thread (Thread ID: 2308) exiting after barrier.
Worker thread (Thread ID: 2267) exiting after barrier.
Worker thread (Thread ID: 2298) exiting after barrier.
Worker thread (Thread ID: 2282) exiting after barrier.
Worker thread (Thread ID: 2313) exiting after barrier.
Worker thread (Thread ID: 2319) exiting after barrier.
Worker thread (Thread ID: 2336) exiting after barrier.
Worker thread (Thread ID: 2339) exiting after barrier.
Worker thread (Thread ID: 2311) exiting after barrier.
Worker thread (Thread ID: 2252) exiting after barrier.
Worker thread (Thread ID: 2331) exiting after barrier.
Worker thread (Thread ID: 2306) exiting after barrier.
Worker thread (Thread ID: 2293) exiting after barrier.

-----STATISTICS-----
Consumers: 100 - Buffer Size: 10
Number of Regular Files: 3116
Number of Directories: 151
Number of FIFO Files: 0
TOTAL BYTES COPIED: 73520554
TOTAL TIME: 14.749 seconds
```