

# **CSE 344 – System Programming Homework 4 Report**

Ahmet Özdemir

In the assignment, I will first describe my path of progress. The game consists of three main files, main.c systemHelper.c and stack.c, and two header files. Now I will explain this programme starting from the main function.

The general flow of the main function is as follows:

- 1 - argument checks
- 2 - buffer memory allocation
- 3 - time initialisation
- 4 - thread creation and thread end waiting blocks
- 5 – adding pthread\_barrier (\* new)
- 6 - statistics information

```

62
63 pthread_mutex_init(&bufferMutex, NULL);
64 pthread_cond_init(&bufferNotEmpty, NULL);
65 pthread_cond_init(&bufferNotFull, NULL);
66 pthread_barrier_init(&barrier, NULL, workersNumbers);
67
68 /**** TIME START ****/
69 gettimeofday(&startTime, NULL);
70
71 dummyControl = pthread_create(&manager, NULL, managerTask, (void *)&directories);
72 errExitSyscall("Error on main(creating thread manager) function", dummyControl);
73
74 for (unsigned int i = 0; i < workersNumber; ++i)
75 { ...
76 }
77
78 dummyControl = pthread_join(manager, NULL);
79 errExitSyscall("Error on joining manager thread", dummyControl);
80
81 for (unsigned int i = 0; i < workersNumber; ++i)
82 { ...
83 }
84
85 gettimeofday(&endTime, NULL);
86
87 /***** TIME END *****/
88
89 long seconds = endTime.tv_sec - startTime.tv_sec;
90 long microseconds = endTime.tv_usec - startTime.tv_usec;
91 double elapsed = seconds + microseconds * 1e-6;
92
93 printf("----- STATISTICS ----- \n");
94 printf("Total files copied: %u \n", filesCopied);
95 printf("Total bytes copied: %u \n", totalBytesCopied);
96 printf("Number of regular files: %u \n", numRegularFiles);
97 printf("Number of FIFOs: %u \n", numFIFOFiles);
98 printf("Number of directories: %u \n", numDirectories);
99 printf("Number of symbolic links: %u \n", numSymbolicFiles);
100 printf("Total time elapsed: %.2f seconds \n", elapsed);
101 printf("----- \n");
102
103 cleanup_resources();
104
105 pthread_mutex_destroy(&bufferMutex);
106 pthread_cond_destroy(&bufferNotEmpty);
107 pthread_cond_destroy(&bufferNotFull);
108 pthread_barrier_destroy(&barrier);
109
110
111
112

```

The manager thread is run first. Here paths are set and source path is opened, then copy files are created in destination path. Only necessary files are created, these files are opened with open on both sides, file descriptors and file paths are written to a struct object named fileBody. This object is also written to buffer. In these intervals, mutex locking operations are also performed where necessary.

```

49 void * managerTask(void * argument)
50 {
51     DirPaths * initialDirs = (DirPaths *)argument;
52     StackNode * stack = createStackNode(*initialDirs);
53
54     while (!isStackEmpty(stack) && !killSignal)
55     {
56         DirPaths currentDirs = pop(&stack);
57
58         const char * sourcePath = currentDirs.sourceDirPath;
59         const char * destinPath = currentDirs.destinDirPath;
60
61         DIR * sourceDir = opendir(sourcePath);
62
63         if (!sourceDir)
64         {
65             //
66         }
67         struct dirent * dEntry;
68
69         while ((dEntry = readdir(sourceDir)) != NULL)
70         {
71             if (strcmp(dEntry->d_name, ".") == 0 || strcmp(dEntry->d_name, "..") == 0)
72             {
73                 //
74             }
75             char sourceFileName[NAME_SIZE];
76             char destinFileName[NAME_SIZE];
77
78             snprintf(sourceFileName, NAME_SIZE, "%s/%s", sourcePath, dEntry->d_name);
79             snprintf(destinFileName, NAME_SIZE, "%s/%s", destinPath, dEntry->d_name);
80
81             if (strncmp(destinPath, sourceFileName, strlen(destinPath)) == 0)
82             {
83                 //
84             }
85
86             if (dEntry->d_type == DT_REG || dEntry->d_type == DT_FIFO || dEntry->d_type == DT_LNK)
87             {
88                 //
89             }
90
91             else if (dEntry->d_type == DT_DIR)
92             {
93                 //
94             }
95         }
96         closedir(sourceDir);
97     }
98     pthread_mutex_lock(&bufferMutex);
99     done = 1;
100     pthread_cond_broadcast(&bufferNotEmpty);
101     pthread_mutex_unlock(&bufferMutex);
102     clearStack(&stack);
103     pthread_exit(0);
104 }
105
106
107

```

If `dEntry->d_type == DT_DIR`, i.e. the found element is a directory, they are stored in a stack data structure. Sub-directories are then called sequentially with a recursive call logic.

Stack data-structures and necessary functions:

```
9
10  typedef struct StackNode
11  {
12      DirPaths dirPaths;
13      struct StackNode * next;
14  }
15  StackNode;
16
17  StackNode * createStackNode(DirPaths dirPaths);
18  DirPaths pop(StackNode ** stack);
19  void push(StackNode ** stack, DirPaths dirPaths);
20  void clearStack();
21  int isEmpty(StackNode * stack);
22
```

General flow of the Worker thread:

```
202
203  void * workerTask(void * argument)
204  {
205      pthread_barrier_wait(&barrier);
206
207      while (TRUE)
208      {
209
210      }
211      pthread_exit(0);
212  }
213
214
215
```

Outputs of tests:

Test1:

```
ahmete@ahmete-Inspiron-14-5401: ~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here
ahmete@ahmete-Inspiron-14-5401:~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here$ make clean
ahmete@ahmete-Inspiron-14-5401:~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here$ make
ahmete@ahmete-Inspiron-14-5401:~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here$ valgrind ./MwCp 10 10 ../testdir/src/libvterm ../toco
py
==10701== Memcheck, a memory error detector
==10701== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10701== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==10701== Command: ./MwCp 10 10 ../testdir/src/libvterm ../toco
py
==10701==
----- STATISTICS -----
Total files copied: 194
Total bytes copied: 25009680
Number of regular files: 194
Number of FIFOs: 0
Number of directories: 7
Number of symbolic links: 0
Total time elapsed: 0.53 seconds
-----
==10701==
==10701== HEAP SUMMARY:
==10701==   in use at exit: 0 bytes in 0 blocks
==10701==   total heap usage: 36 allocs, 36 frees, 344,518 bytes allocated
==10701==
==10701== All heap blocks were freed -- no leaks are possible
==10701==
==10701== For lists of detected and suppressed errors, rerun with: -s
==10701== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ahmete@ahmete-Inspiron-14-5401:~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here$
```

Test2:

```
ahmete@ahmete-Inspiron-14-5401: ~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here
ahmete@ahmete-Inspiron-14-5401:~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here$ ./MwCp 10 4 ../testdir/src/libvterm/src ../toco
py
----- STATISTICS -----
Total files copied: 140
Total bytes copied: 24873082
Number of regular files: 140
Number of FIFOs: 0
Number of directories: 2
Number of symbolic links: 0
Total time elapsed: 0.06 seconds
-----
ahmete@ahmete-Inspiron-14-5401:~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here$
```

Test3:

```
ahmete@ahmete-Inspiron-14-5401: ~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here
ahmete@ahmete-Inspiron-14-5401:~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here$ ./MwCp 10 10 ../testdir ../toco
py
----- STATISTICS -----
Total files copied: 3116
Total bytes copied: 73520554
Number of regular files: 3116
Number of FIFOs: 0
Number of directories: 151
Number of symbolic links: 0
Total time elapsed: 0.07 seconds
-----
ahmete@ahmete-Inspiron-14-5401:~/DERSLER/3_SINIF/Spring/System-Programming/hw4test/hw4test/put_your_codes_here$
```