Gebze Technical University
Department of Computer Engineering - CSE344 System Programming
Spring 2018 - 2019, HW#6 Report
Ahmet Yasir Nacak – 161044042

In this homework, I created a copying utility that copies all contents of a directory to another directory while using threads to actual copying. If we made this program a single thread program, it would take a lot of time to copy a directory so that's why we are using a worker thread pool instead of a single thread. But to make threads worth, we need to decide how many threads we should create for copying. There is also the problem of producer buffer size in this problem. Since the total number of copying instructions that can be in our buffer is also limited, we need to optimize on two factors. To analyze this, I tested the all possible four edge scenarios. The testing directory looks like the following:

```
cse312@ubuntu:~/Desktop/Sys/CSE344/HW06$ du -a A
4       A/B/another_one.mp3
0       A/B/beware
4       A/B/dust2.tga
8       A/B/D/big_file.bak
12      A/B/D
4       A/B/bites_the.wav
28      A/B
5076    A/very_big_doc.pdf
0       A/somefifo
4       A/test.txt
4       A/myboy.jpeg
4       A/C/squirtle.pu
4       A/C/senNeDun
16      A/C/truncate
0       A/C/otherfifo
28      A/C
5144    A
```

First of the cases is where we have a one sized buffer and a single consumer approach. With this approach, I got around 4.6 seconds of copying time. This is the slowest time possible because it has the same running time with a single threaded program.

Second case is where we have our buffer with same size as the files to be copied, N. The buffer can get higher than that, but it does not influence the performance. With this approach, producer fills the buffer beforehand and exits. After that, a single consumer picks all entries one by one and copies all files. This is slightly better than the first approach but since producer is already a fast program, we can not count this improvement as a good one. I got 4.6 seconds of copying time with this approach as well.

Third approach is where we have a single buffer size and N consumers where N is the number of files to be copied. This approach provides us a great improvement over the first two approaches. Since every time producer puts an item into the buffer, a consumer takes it from there and does the copy. This approach immediately makes the performance better because all copying operations run in parallel. I got 3.8 seconds of copying time from this approach.

Final approach is where we have N sized buffer and N consumers. This final approach provides us the most performant result because our producer gets its job done earliest and can exit gracefully. After that, all N consumers do their copies and has the guarantee of quitting right after their copy is done. I got 3.68 seconds of copying from this approach.