Abdulellah Abualshour | RUID: 158006704 | netid: aha59

Marshal Nink | RUID:158009719 | netid: mrn61

Assignment 2: Procs vs Threads

The main purpose of this assignment is to implement a program that is able to compress a piece of text fetched from a specific file and then split this text after compression into several files, depending on user input. However, the split process will not be at the end of the compression, but during that process.

The program will determine the starting index of the compression, so it will start compressing the text at that index and will end at the ending length point, and then the index will be incremented for the next piece to be compressed by the second process. At the end of every process, the compressed string will be saved into a file in the correct order of spawned processes. This program works successfully in parallel; all processes work at the same time to achieve the final stage of compression.

Program Behavior:

The program handles different approaches in a specific way. In case of any error, it will terminate. For example, the program will terminate if:

- No text was in the file.
- Number of parts required is more than number of characters in file.
- Compressed text file could not be created.
- Original text file could not be opened.

Otherwise it will run correctly and compress. It will also override the files in the directory if they existed previously.

Usage:

We were informed that the graders will use a binary of their own to test our code. You may do so. We made sure that the functions compressT_LOLS and compressR_LOLS match the specification in the Assignment2 pdf.

However, if the user wants to test it by themselves, they may modify the main functions inside the source code and put the desired number of parts or file name there, as shown below:

Threads:

```
int main() {

    //Here you should modify to what the number of parts you want is:

    compressT_LOLS(bigfile, 3);

    return 0;

}
```

Processes:

```
int main() {
	//Here you should modify to what the number of parts you want is:
	compressR_LOLS(bigfile, 3);
	return 0;
}
```