Assignment 1: Parallel File Compression utility

Parallelizing the compression of multiple files can significantly enhance performance by leveraging the capabilities of modern multi-core processors. Understanding parallel computing concepts is crucial in today's computing landscape, and this exercise provides a practical opportunity to explore these concepts in the context of file compression.

Parallelizing file compression is categorized as program-level parallelism, involving independent parallel tasks. This form of parallelism is straightforward to handle as it does not necessitate special attention to data sharing. Nonetheless, creating processes is a resource-intensive operation that could potentially impact the efficiency of parallelism. To mitigate this, it is advisable to focus on longer task running times. Therefore, it is recommended to consider working with files of at least 30MB in size

We suggest to start with a sequential implementation that compress one data file at one. Then duplicate your data file and create a version of your code that process two compressions in parallel. Compare resulting timing in sequential and parallel compression of both files.

Now you are ready to start your heavy parallelizing, supposing you have a list of N files stored in an array by their name (see bottom notes). Write a function that parallelize the compression of N file while inspecting different alternative of parallelization (write each in a different function):

- Creating N parallel tasks at same time to process the N files compressions.
- Creating a NB_CORES tasks each time to process compression of NB_CORES files, and when finished start a new set of NB_CORES tasks again and again till finishing the N files.
- Creating just NB_CORES tasks at all, and assign multiple files compression for each task (as equally as possible).
    - Notes: 1) NB_CORES is the number of cores on your CPU; 2) we suggest working on a number N of about 300 files (duplicates).

Performance Analysis:
Investigate the impact of your strategies on the efficiently of parallel compression, by comparing pure sequential execution with 2,3,4..NB_CORES execution, then with the 3 mentioned strategy (draw an excel graph to show scale up of performance).

Notes: We have supplied you with a utility file (utils.c) that includes a function for retrieving a list of files in a designated folder and another function for determining the number of cores on your computer. Additionally, a simple_main.c file is provided to demonstrate the usage of these functions (headers.h is utilized to declare external functions). To compile the provided files, execute the following command: **gcc simple_main.c utils.c**. For testing purposes, run: **./a.out folderName** (replace folderName with the folder containing the files for compressions).