Brandon Smith and Anthony Chou
Asst2: Processes vs Threads (Round 0)

Readme

compressT_LOLS.c

Usage: compressT_LOLS.c filename parts

This file is the thread version of LOLS. The file is self contained and run without needing any worker programs. The program takes only two arguments which are the filaneme of the unencoded file and the number of parts the unencoded file should be split into. If the number of parts received is invalid or 0 the program will exit. If the number of parts received is greater than the number of characters in the file, the program will reduce the number of threads to the max number of characters. If the file is empty, the program will also exit. The threads version will encode the given file into the amount of parts specified using LOLS. Each file will be made on a separate thread. The way the unencoded file is divided is each part is of size "length of unencoded file" / "parts". If "length of unencoded file" % "parts" is not 0. That value is added onto the first part.

compressR_LOLS.c

Usage: compressR_LOLS.c filename parts

This file is the process/child version of LOLS. The file relies on a worker binary which is called "compress_R_worker_LOLS". If that binary is not found the program will exit. This binary can be compiled using the c file "compress_R_worker_LOLS.c". An explanation for my method of encoding will be explained with compressR_worker_LOLS.c. The process and thread version have the same expected behaviors regarding splitting and errors.

compressR_worker_LOLS.c
This is the worker for the process version of LOLS. It needs to be compiled to "compressR_worker_LOLS" for the process version to run. This file has a main method which takes in a char pointer array with the arguments. Both the threaded and process version use the same method for encoding so this will be described here. The encoding reads the file character by character. If the file is not found the program will exit. The encoding takes in a filename, length to encode, starting, and part numbers. I ignore non alphabetic characters. If the file cannot be written to the program will exit. This program will attempt to overwrite files without warning. There is no check for existing files. Output file will be <filename>_LOLS<part number> except if the number of parts is 1 where there will not be a number appended to the end.