CS 214: Systems Programming, Fall 2016
Assignment 2: Procs vs Threads (round 0)

readme.pdf

Jeffrey Huang, jh1127
Richard Li, rl606

Overall Design:

Our design is an iterative string compressor that only compresses characters and removes all non-alphabetic characters during compression. After retrieving all the information from the file that the user has provided, the program iterates through the string, splits it into the number of parts that is going to be compressed, then sent to an encoding method (called compress) to be compressed and printed into a file. If the user decides that they need 100 parts of compression for a 19 character file

Thread Design (compressT_LOLS.c):

The thread design utilizes threads to optimize the speed of the program. Instead of having a single path where the program iterates through the entire string and splices at the necessary places, the main method initializes by finding the indices at where the string needs to be split. These split locations are then fed to a thread where the information is to be processes individually. The thread only contains the method where the information is going to get processed. Hence the information that is passed will be pipelined with no memory overlap so that the information passed down will be handled and returned properly. Information that is passed are 3 arguments, all as integers.

1. the output file number
2. the index of which the string is to be spliced
3. the length of the string that is spliced.

As a result, if the string needs to be split into 2 parts, there will be two threads created to handle both parts of the split string. The compression prints to the output file that is generated by the program under the file extension where the period is replaced with an underpart and postpend '_LOLS' and the part number to notify the user that the part has been compressed in order.

Compile command:
gcc -o compressT_LOLS compressT_LOLS.c -pthread

Run command:
./compressT_LOLS file.txt 5

Process Design (compressR_LOLS.c):

The multiprocessing design spawns many processes to concurrently compress the string. The program first determines how many child processes to spawn, then creates that many processes using fork() and execv(). The child processes are passed 4 arguments, all as strings.

1. Name of file to be compressed
2. Offset in file of where to start compression
3. Length to compress
4. Process number

The child process is initiated by execv, passing the above arguments. Each child process skips to the offset indicated in the arguments, and compresses the length specified in the arguments, writing to a new output file in the format specified in the instructions.

Compile command:
gcc -o compressR_worker_LOLS compress_worker_LOLS.c
gcc -o compressR_LOLS compressR_LOLS.c

Run command:
./compressR_LOLS file.txt 5