Message Descrambling

Dr. Christopher Marcotte — *Durham University*

Message descrambling

The fundamental principle of coded messages is the required use of external information in order to turn something superficially information-sparse into something information-dense. One of the most basic ciphers is called the Caesar cipher, which circularly shifts a letter in the original message in the alphabet by a fixed amount.

The following is a scrambled message:

> *Zpv!uijol!zpvs!qbjo!boe!zpvs!ifbsucsfbl!bsf!voqsfdfefoufe!jo!uif!ijtupsz!pg!uif!xpsme-!cvu!uifo!zpv!sfbe/*

And `encoder.c` was used to scramble it:

encoder.c

```c
#include <stdio.h>  // Import the printf function
#include <omp.h>    // Import the OpenMP library functions

// Start of program
int main(){

   // This is a "character array", consisting of 105 letters in byte form.
   char mymessage[] = "This is a placeholder of the original message";

   // The message is encoded by shifting all letters by one, i.e.
   // A becomes B, c becomes d and so forth. This shift is easy to implement
   // in byte form as all you need to do is to add 1 to a character.
   // To printf a single character, use e.g.
   //       printf("%c\n", mymessage[0]);
   // To printf the whole array do e.g.   (subtle difference between %c and %s)
   //       printf("%s\n", mymessage);

   /* MISSING CODE: for loop over the 105 characters and adding the value 1 to each entry; */

   printf("%s\n", mymessage); // This prints a character array to screen

   return 0; // Exit program
}
```

We will use this encoder to investigate parallel decryption of the secret message.

Exercise

Fix and finish the serial encoder. Include a way to determine the length of `char mymessage[]` programmatically.

Exercise

Write a serial decoder which undoes the action of the above program. What is the decoded message? *Read through the comments in the code*!

Hint

You may find it easier to finish this exercise *after* completing the OpenMP Thread Test exercise.

Exercise

Parallelize the decoder using OpenMP. Use suitable pragmas and run the decryption with multiple threads. Insert a statement in the for-loop that prints the current thread number and decryption result as the decoder progresses. Run the code repeatedly and vary the number of threads (`OMP_NUM_THREADS`); What do you observe?

Exercise

Write a new encoder/decoder pair which only produces the correct message if the number of threads used by the sender (the encoder) is the same as the number used by the receiver (decoder).

Exercise

You may notice that such a short text is not sufficient to really tax even a single thread, nevermind several OpenMP threads. Encode the plain text version of *Frankenstein; Or, The Modern Prometheus* by Mary Wollstonecraft Shelley. Time the encoding with several threads, and produce a plot showing the speedup from using multiple threads.

Warning

If you are using a Windows terminal without Unicode support, or a font lacking unicode support, you *may* need to normalize the encoding of the text file to ASCII, which is a subset of UTF-8, the nigh-universal encoding for text, now. The unfortunate thing about UTF-8 for us is that it is variable-width, 1 – 3 bytes per character, which will make our lives more difficult for no good pedagogical reason. You can create an ASCII-encoded version of the Frankenstein file with the command:

```
iconv -f UTF-8 -t ASCII Frankenstein.txt > ASCII-Frankenstein.txt
```

which may solve some subtle issues around encoding characters coherently.

Additionally, there is no portable method to determine the length of a text file in *C* – meaning to get an accurate character count, we would need `fstat` by invoking `#include <sys/stat.h>`. However, you may not *need* to do this to complete the exercise – a quick and (technically incorrect) approach will likely work fine, if not robustly, using `fseek` and `ftell`.

Aims

- Review of some basic *C* programming patterns.
- Familiarity with simple usage of OpenMP library functions.
- A gentle introduction to simple loop parallelism in a novel setting.