CS35L_Lab_5

February 16, 2018

1 Buffered versus unbuffered I/O

1.1 Objective

For this laboratory, we will implement transliteration programs `tr2b` and `tr2u` that use buffered and unbuffered I/O respectively, and compare the resulting implementations and performance.

1.2 Specification

Each implementation will be a **main program** that takes **two** operands `from` and `to` that are **byte strings** of the same length, according to which the program will convert every byte of standard input in `from` to the byte with the same index in `to`; then it will output the result to standard output.

The implementations will report an error if `from` and `to` are not of the same length, or if `from` has duplicate bytes.

To summarize, the implementations will act like the standard utility `tr`, except that they have no options; characters like [, - and \ have no special meaning in the operands; operand errors will be diagnosed; the implementations act on bytes rather than on (possibly multibyte) characters.

1.2.1 Background

A **byte string** is in essence the internal representation of string as it is stored in the computer. The commonly known UTF-8, US-ASCII, and Unicode are all **character encoding system** that converts the corresponding string and byte string from and to each other. Depending on the system, the same string can be encoded into different byte strings; similarly, the same byte string representation can be decoded into different strings. When we see garbage characters or Mojibake when opening a file, it is usually because the file is decoded with the unintended character encoding.

Here is an excellent answer from Stackoverflow that explains this concept very well.

1.3 Lab Log

1.3.1 Create the program: tr2u, tr2b

1. Because the use of hashtable can significantly speed up the two programs, I just tried to look for a library for hashtable in C.

2. Unfortunately, there does not seem to be an implementation of hashtable in the standard library of C. So, we are going to write a rather inefficient algorithm.