

CSI 402 – Spring 2011
Programming Assignment II

Administrative Information

- **Deadline:** 11 PM, Wednesday, Mar. 2, 2011.
Cutoff: 11 PM, Friday, Mar. 4, 2011.
- The program must have two or more C source files.
- All the files (C source files, header files (if any) and the `makefile`) must be submitted together using the `turnin-csi402` command.
- README file
 `~csi402/public/prog2/prog2.README`
will be available by 10 PM on Monday, Feb. 21, 2011.
- The README file will contain information regarding `turnin-csi402` and additional specifications for the `makefile`.

Project Description

Goal: To get some practice in handling binary files.

- Idea based on how Unix represents directory entries.
- Program should do two things:
 - Produce a binary (“unformatted”) file from a text (“formatted”) file and vice versa.
 - Go through a binary file and compute statistical information.

Weightage: 7%

Total Points: 100 (Correctness: 85, Str. & doc: 15).

Form of each line of text file:

string<tab>*integer*

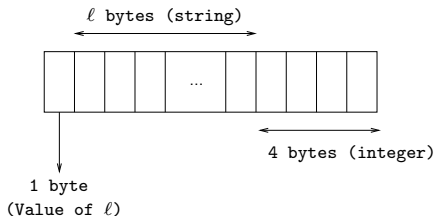
- The string and the integer on a line are separated by *exactly one* tab.
- The string does not contain any whitespace characters.
- The length of the string, denoted by ℓ , satisfies the condition $1 \leq \ell \leq 255$.
- The integer is non-negative and its value can be stored in a variable of type `unsigned int`.

Project Description (continued)

Representation for each line in the binary file:

Line in the text file: `string<tab>integer`

Let ℓ denote the length of the string.



Total length = $\ell + 5$ bytes.

Note: The '`\0`' character of the string is not stored in the binary file.

Project Description (continued)

Example:

Line in Text File -----	Length in binary file -----
input.dat<tab>14721	14 bytes
output.c<tab>12	13 bytes

Thus, for this example, the size of the binary file will be exactly 27 bytes.

Unix Command Line: Two forms are possible.

Unix Command Line – First Form:

% p2 *flag infile outfile*

- p2: Executable version of your program.
- *infile*, *outfile*: Names of input and output files.
- *flag*: May be either `-t` or `-b`.
 - If *flag* is `-t`, then input file is a text file and output must be a binary file (i.e., the program must produce the binary file corresponding to the given text file.)
 - If *flag* is `-b`, then input file is a binary file and output must be a text file (i.e., the program must produce the text file corresponding to the given binary file.)

Unix Command Line – Second Form:

```
% p2 -s infile
```

- *infile*: Name of binary input file.
- The only flag allowed in this case is `-s`.
- In this case, the program should not produce an output file.
- The program must write to `stdout` the following values:
 - 1 The length of a shortest string in the input file.
 - 2 The length of a longest string in the input file.
 - 3 Value of the smallest integer in the input file.
 - 4 Value of the largest integer in the input file.

Assumptions regarding input:

- See the handout.

Errors to be detected:

- Usual command line errors; see the handout.

Suggestions:

I. Converting a text file to a binary file:

- Use `fscanf` to read the string and the integer on each line of the input file.
- Use a variable of type `unsigned char` to store the length of the string on each line of the input file. The length should be written to the output file using `fwrite`.
- Use a variable of type `unsigned int` to store the integer value from each line of the input file. This value should be written to the output file using `fwrite`.

II. Converting a binary file to a text file:

- Use `fread` to read the length of the string, the string itself and the integer from the input (binary) file.
- Be sure to add the `'\0'` character at the end of the string.
- Write to the text file using `fprintf`; be sure to add the tab character (`'\t'`) between the string and the integer.

III. Processing the binary file (-s flag):

- Proceed as though you are converting the binary file to the text file.
- Instead of producing a text file, keep track of maximum & minimum lengths of strings and maximum & minimum integer values.
- When you write to `stdout`, be sure to use `fflush(stdout)`.

Other Suggestions

- Use the Unix `diff` command to check whether the output produced by your program is identical to the outputs of sample test cases. (Examples to show the use of `diff` will be presented in class.)

Note: The grading script will use the `diff` command to check whether your program produces correct outputs when flags `-t` and `-b` are used on the command line.

- Consider the following organization for your C program.
 - A C source file containing the function `main`.
 - A C source file containing functions needed to convert a text file into a binary file.
 - A C source file containing functions needed to convert a binary file into a text file.
 - A C source file containing functions needed to produce outputs for the `-s` flag.