CSI 402 – Systems Programming – Spring 2012

Programming Assignment II

Date given: Feb. 16, 2012 **Due date:** Feb. 27, 2012
Weightage: 5%

The regular deadline for this assignment is **11 PM, Monday, February 27, 2012**. With lateness penalty, the program will be accepted until **11 PM, Wednesday, Feb. 29, 2012**. The assignment will *not* be accepted after 11 PM on Wednesday, February 29, 2012.

Very important: Your source program must consist of two or more C files and you must also have a `makefile`. (Additional information regarding this requirement is given later in this handout.) The C files (with extension "`.c`"), header files (with extension "`.h`") and the `makefile` must be submitted together using the `turnin-csi402` command. Instructions for using `turnin-csi402` and additional specifications for the `makefile` will be included in the `README` file for this assignment.

The total grade for the assignment is 100 points, with 85 points for correctness and 15 points for structure and documentation.

The purpose of the program is to print a specified sequence of bytes from an input file to `stdout` in a specified radix.

The executable version of your program must be named `p2`. (Your `makefile` must ensure this.) The program will be executed by a command line of the following form:

`p2` *infile* *start* *end* *flag*

The interpretation of the command line arguments is as follows.

(a) The argument *infile* represents the name of an input file. The file may be a text file or a binary (i.e., an unformatted) file. Your program must treat the file as just a sequence of bytes. The starting byte position in the file is assumed to be 0.

(b) The parameters *start* and *end* specify respectively the starting and ending byte positions in the file. Both of these values are specified in decimal. When these values are nonnegative, there may be an optional `'+'` sign preceding the integer value. When both *start* and *end* are nonnegative integers that represent valid byte positions in the file and the value of *start* is less than or equal to that of *end*, your program should output to `stdout` all the bytes in positions *start* through *end*. As a special case, the argument *end* may be given as -1. In that case, your program must print all the bytes in the file starting from byte position *start* and ending with the last byte of the file. (Additional specifications regarding the output are discussed on page 2 of this document.)

(c) The *flag* argument may be one of the following four possibilities: `-b`, `-q`, `-o` or `-h`. These four flags require that the specified bytes be output in binary, quarternary (i.e., base 4), octal and hexadecimal respectively.

Example 1: Suppose the command line is the following:

```
p2   file1.dat  230   +257  -b
```

If no error conditions occur, your program must output to `stdout` the bytes in positions 230 through 257 (both inclusive) of the file `file1.dat`, with each byte written as a binary string. (Error conditions to be detected by your program are discussed later in this handout.)

Example 2: Consider the following command line:

```
p2   file2.dat  +174  -1  -o
```

If no error conditions occur, your program must output to `stdout` all the bytes in positions 174 through the last byte of the file `file2.dat`, with each byte written as an octal string.

Additional specifications regarding the output: Recall that the specified bytes must be written to `stdout`. In addition, the output must satisfy the following conditions.

(1) When the flag is `-b`, each byte must be printed as a string of exactly eight bits. (Thus, leading zeros, if any, must also be printed.) Similarly, when the flag is `-q`, `-o` or `-h`, each byte must be printed respectively as as a string of exactly four quarternary digits (0 through 3) or exactly three octal digits (0 through 7) or exactly two hexadecimal digits (0 through 9 and A through F).

(2) Each line of output (except the last line) must begin with the starting byte position value as a decimal integer followed by the character `':'` and one or more spaces. This must be followed by the contents of exactly 10 bytes, with one or more spaces separating successive byte values. The last line also begins with the starting byte position followed by the `':'` character; the only difference is that the last line may contain fewer than 10 bytes. (The idea is that the byte values in successive output lines must line up nicely so that it will be easy for a human being to view and understand the output.)

Examples will be presented in class to illustrate the above requirements.

Your program must detect the following fatal errors. In each case, your program should print an appropriate error message to `stderr` and stop.

(1) The number of command line arguments is not equal to five.

(2) The input file specified on the command line cannot be opened.

(3) The *flag* specified is not one of `-b`, `-q`, `-o` or `-h`.

(4) There are errors in the values of *start* and *end* specified on the command line. The specific errors are as follows. In each of these cases, it is enough to output a short error message of the form "`Invalid range specified.`".

   (i) The value of *start* is not an integer or it is a negative integer.

(ii) The value of *start* is larger than the largest byte position in the file.

(iii) The value of *end* is not an integer or it is a negative integer other than -1.

(iv) The value of *end* is larger than the largest byte position in the file.

(v) The values of *start* and *end* are both nonnegative integers but the value of *start* is larger than that of *end.*

Structural requirements: Your submission must have *at least two* C source files, zero or more header files and a `makefile`. Additional requirements on the C source files are as follows.

(a) One source file must contain just the `main` function.

(b) One or more source files must contain functions that convert a byte into its representation in the required number base (binary, quarternary, etc.).

Information about `README` file: The `README` file for this assignment will be available by 10 PM on Monday, February 20, 2012. The name of the file will be `prog2.README` and it will be in the directory `~csi402/public/prog2` on `itsunix.albany.edu`.