Operating Systems. Home Work #1.

Due to April 8 2017, 23:55.

Part I (data_filter.c)

We say that one byte is "printable" if its value is in [20, 126]. Write a C-program that reads data from an input file, filters out unprintable characters, and writes the remaining printable characters to an output file. The program prints statistics report when complete.

Command line argument:

1. Data amount - a value saying how much data to process. Assume that the string contains an integer number and a letter specifying units. Possible letters are "B" (bytes), "K" (kilobytes), "M" (megabytes), "G" (gigabytes).
   Examples: "5K" – process 5 kilobytes of input, "2G" – 2 gigabytes, "671M" – 671 megabytes.
2. Input file name – a value saying where the data shall be read from.
3. Output file name – target output.

Notes:

1. Print error message and quit for any problem with command line arguments.
2. Collect input data into a buffer before filtering. The size of the buffer shall be dependent on the size of the request. Don't implement neither reading 1 byte in a time, nor "read-them-all-at-once" approach. Allocate something about 0.5-1K for small requests and 2-4K for big requests. Also, don't use "one size fits all" policy.
3. Aggregate data in a buffer before output. Same rules apply as for the input in the previous paragraph.
4. Check all the return codes and print error notifications upon every possible failure.
5. Final report string includes three numbers – number of requested bytes, number of bytes that were read, and the number of the printable characters.
   Example: "461 characters requested, 461 characters read, 288 are printable".
6. If the requested amount is bigger than the file size then return the file descriptor to the beginning of the file and continue iterating through the same data again.
   (Check *lseek* system call, "*man 2 lseek*" in the shell.
   Hint: the beginning of the file is referenced as **SEEK_SET**, and the offset can be 0.)
7. Use **read/write** system calls to process the file content. I.e. **NOT** fread/fwrite/ifstream/ofstream etc.

Part II (data_filter_launcher.sh)

Write a wrapper script that launches data_filter.

1. The script defines 3 environmental variables:
   a) DATA_SIZE – contains amount of data to process. A string of format of command line argument in Part I.
   b) INPUT_DATA – filename data_filter reads the input from.
   c) OUTPUT_DATA – filename data_filter writes the data to.
2. The script executes data_filter passing the values of the variables as the command line arguments.

Notes:

1. The script file shall be a self-contained executable script. I.e. the first line of the script file shall be "#!/bin/bash".
2. INPUT/OUTPUT_DATA filenames should contain full paths.
3. Check your code with INPUT_DATA equals to "/dev/urandom".

Submission guidelines

1. Submit one zip archive. The name of the file is ex1_012345678.zip, where "012345678" is substituted by your ID number (מספר תעודת הזהות).
2. The zip archive contains just two files: data_filter.c, data_filter_launcher.sh. Same names for everyone.
3. Submission is in Moodle.