Operating Systems(PG)
Monsoon 2016
Assignment 1
Deadline: Friday 26 August 2016 11:55 PM

Date: 21 August 2016

Objective : Learn about system calls

Part 1: tr command
Write a program to simulate UNIX command 'tr'

The syntax of tr command is:

```
        $ tr [OPTION]  [SET1]  [SET2] < [Input File] > [output file]
```

Note: 1)If input file or output file is missing then you have to use standard input and output for it.
2) create the output file path if it doesn't exist.

Options to implement:

-d : delete characters in SET1, do not translate.

-s : replace each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character.

-c : use the complement of set 1.

Examples of tr command:

```
1) $  tr 'abc' 'def'
> abcd
defd
```

2) tr [:upper:] [:lower:] < ../../ip.txt > op.txt
It will read text from ip.txt and translate all uppercase characters to lowercase characters and write it in op.txt

3) tr -d 'abc' < ip.txt
It will read text from ip.txt and delete the occurence of characters in SET1 and print it on standard output.

4) tr -cd [:digit:]
Remove all characters except digits.

SET can have range queries also.

```
$ tr 'a-c' 'd-f'
```

Input Format:

1) ubuntu:~$ ./a.out abc def
```
>abcd
defd
```

2) ubuntu:~$ ./a.out [:upper:] [:lower:] -I ../../ip.txt -O op.txt
It will read text from ip.txt and translate all uppercase characters to lowercase characters and write it in op.txt

3) ubuntu:~$ ./a.out -d 'abc' -I ip.txt
It will read text from ip.txt and delete the occurence of characters in SET1 and print it on standard output.

4) ubuntu:~$ ./a.out -cd [:digit:]
Remove all characters except digits.

NOTE:
To specify input file use -I option (NOT <)
To specify output file use -O option (NOT >)

Handle only [:lower:], [:upper:],[:digit:],[:space:][:punct:]
Handle error cases and print appropriate messages.
You should handle all combinations of the options.

PART 2: split+tac command

Write a program to split all the files in the given source directory, based on number of lines specified similar to shell command "split -l <n>" and copy them to the destination directory (create the path if it does not exist) by reversing the lines in each of the splitted files similar to shell command "tac <filename>".

Input format:
ubuntu:~$ ./a.out <source dir> <path to destination dir> <#lines>
source dir, path and #lines will be given through command line arguments only!

Examples:
ubuntu:~$ ./a.out Assignment/ dir1/dir2/dest_dir 20
Here Assignment is the source directory. All files in this directory has to be split into

chunks containing 20 lines each. dest_dir will now contain these chunk files with their lines in reverse order. The path dir1/dir2 need not exist. Check accordingly and create if it does not exist.

a. Input structure:
Consider the following structure for source directory ‘Assignment’.

```
Assignment
        |_ abc
        |_ jkl
```

b. Output structure:
Your dest_dir structure should contain the chunks as follows:

```
dir1
  |_dir2
      |_dest_dir
        |_ abc_1
        |_ abc_2
        |_ ....
        |_ abc_n
        |_ jkl_1
        |_ jkl_2
        |_ ....
        |_ jkl_m
```

Now, abc_1 will contain the lines 1- 20 from abc in reverse order(20,19...1).
Similarly for other chunks.
Note : All the files in source directory would be text files only.

PART 3: Word count
Write a program to simulate “wc” command.

Options to be implemented:
-l : Number of lines
-c : Number of bytes
-m : Number of characters
-L : Length of longest line
-w : Number of words

Input format :

```
ubuntu:~$ ./a.out -lc abc.txt
ubuntu:~$ ./a.out -lmc /home/user/Desktop/Assignment/abc.txt
ubuntu:~$ ./a.out -c -L ../../Assignment/abc.txt
ubuntu:~$ ./a.out -w Assignment/abc.txt
ubuntu:~$ ./a.out abc.txt
```

All options and file name will be given through command line arguments only. You should handle all combinations of the options.

Output format :
Output should be similar to the output provided by "wc" command in shell.

Useful System Calls :

- read
- write
- open
- close
- mkdir
- access
- scandir
- readdir
- closedir

General Guidelines :

- You are not supposed to use STLs or 'system' library function of linux. If found violated, your submission will not be evaluated.
- Please use above mentioned system calls only. Do not use wrappers.
- Indent and comment the code properly.
- Your name and roll number should be included as comments at the beginning of code.
- Due credit will be given to modularity of code.
- ZERO tolerance towards any kind of code plagiarism.
- Strictly follow the upload format and deadlines. All invalid submissions will not be considered for evaluation.
- Make sure you do not upload any executables.
- Your programs should be scalable for large input files.
- Handle error cases wherever required.
- Refer to the MAN page of the command for more understanding.
- **You can use string.h header file but you are not allowed to use string stl of cpp**

Upload format:

Create a folder named your roll number(20XXXXXXX_AssignmentNo).
Inside that create three folders named Part1, Part2 and Part3
In each folder, place your '.c' or '.cpp' files.

Create a tar.gz of the above folder(20XXXXXXX_AssignmentNo) named “<RollNo>_AssignmentNo.tar.gz” and upload it.

Example: 2015123433_Assignment1.tar.gz