EE442 HOMEWORK 1

UNIX Commands

Due: March 26, 2017, 23:55

A. Doğa Hakyemez ARC-201 dogahakyemez@gmail.com
Okan Çalış   calis.okan@metu.edu.tr
Çağrı Uslu

Part 1: `ls442` (75 pts)

In this part, you are asked to implement the UNIX command `ls` in C. `ls` is used to list files in a directory with different sorting and information options. You can learn more about `ls` with `man ls` and `info ls` commands. Your program should be named `ls442`.

Specifications:

- Without arguments `ls442`, should work as `ls -l`. The list should be sorted alphabetically (by default). Files starting with '.' should not be printed.
- In one line there should be:
    - File type and permission bits. The first character should be 'd' for directories; for other types you can use '-'. The next 9 characters represent permissions given to the user, the group, and the others in groups of 3 characters. If reading is permitted to a party, the corresponding character should be 'r', otherwise '-'. This is similar for writing ('w') and execution ('x'). You can ignore other cases.
    - The number of hard links
    - Owner name
    - Group name
    - Size
    - Modification time
    - File name
- The program should work with any permutation of the options. The desired options are as follows:
    - -a: List all files and directories including those starting with '.'
    - -g: Do not display owner name
    - -G: Do not display group name
    - -i: Print inode number before the file name
    - -S: Sort by size (largest first, files with the same size are to be sorted alphabetically)
    - -U: Do not sort; list the files in the order they are stored in the directory
    - -r: Reverse the sorting order (e.g. smallest first for -S)
    - -Q: Enclose file names with double quotes
- You can ignore total number of blocks, which is printed in the first line.
- Do not assume anything about the number of files in a directory.
- Your program output does not have to be colored.

- You can assume files and directories only have ASCII characters.
- Use only C POSIX library (but you can use GNU extensions in the said libraries). The full list is available at http://pubs.opengroup.org/onlinepubs/9699919799/idx/head.html
- You should compile your code with GCC (GNU Compiler Collection).

Hints:

- While you can use an IDE like Code::Blocks or Eclipse, you can also write your program in a text editor, such as Sublime Text or Geany.
- You can find more information about headers, functions, and some usage examples at: http://pubs.opengroup.org/onlinepubs/9699919799/
- To compile your program, you can use the command: `gcc ls442.c -o ls442`

Part 2: String pattern search with a shell script (25 pts)

In this part, you are asked to write a shell script that searches an input text file for lines matching a pattern string and prints the results into an output text file. You are expected to use the 'grep' utility in your script.
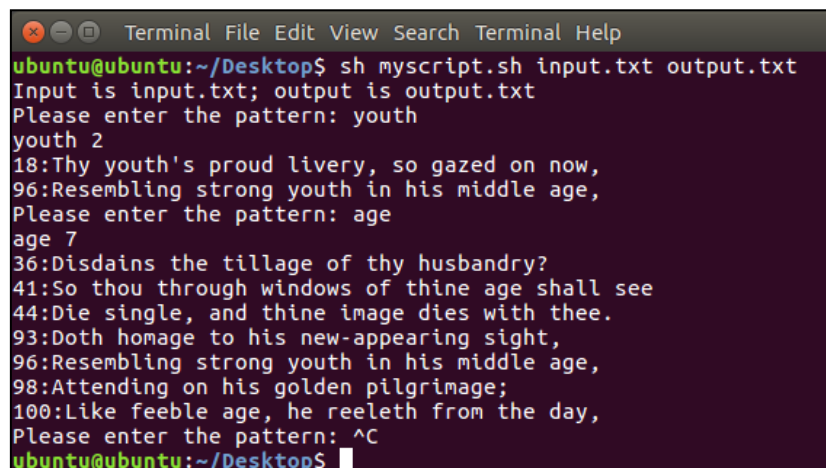
Specifications:

1) The script must be called with two arguments: the input and output text files, respectively.
2) Once called, print the entered input and output file names to the shell.
3) Then, print "Please enter the pattern: " and wait for user input.
4) After the user enters a pattern, print the results on the shell and append them to the output text file in the following format:

Shell output format:

PATTERN COUNT_OF_MATCHING_LINES
LINE_NUMBER: LINE
Example:
```
youth 2
18:Thy youth's proud livery, so gazed on now,
96:Resembling strong youth in his middle age,
```

Output text file format:

> DATE INPUT_FILE PATTERN COUNT_OF_MATCHING_LINES { LIST_OF_LINE_NUMBERS }
> **Example:**
> `Fri Mar 17 01:59:10 +03 2017 input.txt youth 2 { 18 96 }`

5) Go back to step 3. The script must keep requesting input until the user exits by hitting *Ctrl+C*.

Execution:

Then, use your script to obtain how many lines in the provided text file contain the following regular expressions:

- `thou`
- `Thou`
- `[Tt]hou`   (lines containing either "Thou" or "thou")
- `[Tt]h..`   (lines containing a four-character string that starts with "Th" or "th")
- `^But`   (lines starting with "But")
- `ee$`   (lines ending in "ee")
- `^$`   (empty lines)

The script call will be in the form: `sh myscript.sh input.txt output.txt` (from bash terminal) or `./myscript.sh input.txt output.txt` (from bourne shell).

Input Text:

The input text (`shakespeare.txt`) consists of ten sonnets by William Shakespeare, each separated by an empty line.

Remarks:

1. You should insert comments to your code at appropriate places without including any unnecessary detail. <u>Comments will be graded</u>. You have to write to-the-point comments in your code, otherwise it would be very difficult to understand. If your output is wrong, the only way we can grade your homework is through your comments.

2. Send your homework compressed in an archive file with the name "e<student_ID>_HW1" (e.g. e1234567_HW1.tar.gz). The archive file should include:

   - For part 1: your **source file(s)** and **header file(s)** (if they exist)

   - For part 2: your **shell script** and **output text file**

3. Your work will be graded on its correctness, efficiency and clarity as a whole.

4. Late submissions are welcome, but penalized according to the following policy:

   - 1 day late submission: HW will be evaluated out of 70.

   - 2 days late submission: HW will be evaluated out of 50.

   - 3 days late submission: HW will be evaluated out of 30.

   - 4 or more days late submission: HW will not be evaluated.

Good Luck!