MYY106 - Introduction to Computer Science
5th lab

Filters

In this lab we will create command chains to process the results of commands with various filters. As we did last time, we will use exclusively terminal commands (we will not open `Files` or `pluma`).

Open a terminal, go to the directory `myy106`, create a new directory `lab5` there, and go inside it. Create (with the text editor `vi`) the file `answers-lab5.txt` and write your name and ID in it.

1. Use the command `wget` to download the file `chat.log` to the `lab5` directory from the address `https://www.cse.uoi.gr/~myy106/files/chat.log`

   Record the command you used in the answers file.

2. First display the contents of the file `chat.log` on the screen, and then select the appropriate filter according to queries (α′)-(θ′), which will take as input the output of the previous query command and **display the result on the screen**.

   Hint: Use the '|' (pipe) to combine filters and create a chain of commands. At each step you can use the up arrow to invoke the previous command, and add the new filter at the end.

   (α′) Display the lines of the previous command in reverse order.
   (β′) Add the line number in front of each line.
   (γ′) Show only the first 10 lines.
   (δ′) Select to display only the 2nd column (field).
   (ε′) Convert all letters to lowercase.
   (στ′) Sort the lines in alphabetical order.
   (ζ′) Remove duplicate lines.
   (η′) Show only lines containing the word "error".
   (θ′) Display the number of lines containing the word "error".

Packing and compression

3. To transfer many files together (e.g. to a flash drive or over a network), we pack them into a file with the `tar` command and compress them with `gzip`. This reduces their size and speeds up copying, while it is more practical to transfer a single file instead of many separate ones.

   (α′) Use the `ls` command (with the appropriate parameter) to see the details of the `chat.log` file. Record only the file size in the answers file.
   (β′) See the contents of the file with the command `cat` (or `more`, or `less`).
   (γ′) Compress the file `chat.log` with the command `gzip`. Find the size of the compressed file, and record it in the answers file.
   (δ′) Try to view the contents of the file again with the same command you used above. What do you notice? Record your comment in the response file.
   (ε′) Find the appropriate command to properly view the contents of the compressed file. Record the command in the answers file.
   (στ′) Unzip the compressed file `chat.log.gz` again using the appropriate parameter to preserve the input file. Record the command you gave in the response file.
   (ζ′) Go to your personal directory, and create a copy of the directory `myy106` with the name `new_myy106`. With the `tar` command "pack" the entire directory `new_myy106` to create the file `new_myy106.tar` and use the appropriate parameter to display information about the packing. Record the command you gave in the answers file.

(η′) Compress the file `new_myy106.tar` and record the command you gave in the answers file.

(θ′) Use a single command, with the appropriate parameter, to decompress and unpack the `new_myy106.tar.gz` file and record the command you gave in the answers file.

Links to files

4. Links (called shortcuts in Windows) are "pointers" to other files, and are used to access a file from different directories, without having multiple copies.

   (α′) Create a symbolic link (soft link) for the file `chat.log` with the name `link`.

   (β′) Create a hard link to the file `chat.log` with the name `hard`. Notice that, looking at the detailed contents of the directory, the number before your username has changed to 2 for these files.

   (γ′) View the files contained in the current directory (with details) and notice how the links you made appear.

   (δ′) Use the `file` command to find out what kind of file `link` is. Do the same for `hard`, and `chat.log`.

   (ε′) Delete the file `chat.log`. What happens to the `hard` link? Try viewing its contents.

   (στ′) What happens to `link`? Can you read its contents? View the contents of the directory again, and notice how the link appears. Delete `link` as well.

   Record your conclusion about how links work in the answers file.
   (e.g. "When the original file is deleted ...")

Wildcards and regular expressions

For the next questions, we will download the file `https://www.cse.uoi.gr/~myy106/files/lab5.tar.gz` (with the `wget` command). We will decompress this file (with the `gunzip` command) and unpack it (with the `tar` command). A directory named `lab5-files` will be created, which will contain various files, some of them hidden.

5. Enter the directory `lab5-files` and view its contents (including the hidden files).

6. To understand how ***wildcards*** work, issue the following commands and observe why the files you see match in each case. Record in the answers file what the wildcards matched.

   (α′) `echo *`

   (β′) `echo \*`

   (γ′) `echo .*`

   (δ′) `echo [0-9]*`

   (ε′) `echo ?`

   (στ′) `rm a` (delete the file `a` which is the only one with a letter in the name, and then try the previous command again)

   (ζ′) `echo ?` (What do you notice?)

When we want to use a wildcard character with its literal meaning, we put the ***escape character*** '`\`' in front of it.

When wildcard characters appear in a command, the shell replaces them with the matching file names before passing them as arguments to the command.

However, if the command itself uses wildcard characters (e.g. `find` or `grep` within a command chain), we need to protect them from the shell, so that they are not replaced. This is done with single or double quotes (`' '` or `" "`). The difference is that with double quotes the shell replaces the variables (e.g. `$USER`, `$HOME`), while with single quotes it does not, e.g.:

```
$ echo "$(date) and 2+2=$((2+2))"
Mon Nov 26 13:04:39 EET 2019 and 2+2=4
$ echo '$(date) and 2+2=$((2+2))'
$(date) and 2+2=$((2+2))
```

7. Use `ls` to see the files in the directory that start with `'a'`.
8. See the files that start with `'a'` and end with `'g'`.
9. See the files that have numbers organized like a date, that is, starting with numbers of the format `YYYY_MM_DD`).

Extended regular expressions

10. To understand how extended regular expressions work, we will issue the following commands in the `lab5-files` directory and observe which files are included in the results each time.

    (α′) `grep -E 'colou?r' test`

    (β′) `grep -E 'favou*r' test`

    (γ′) `grep -E 'kostas|John|Alice' test`

    (δ′) `grep -E '^[0-9]' test`

    (ε′) `grep -E '^[0-9]*' test`
    Why do all lines match and not just those with numbers?

    (στ′) `grep -E '^[0-9]+' test`
    What has changed now?

11. Now try some regular expressions with `grep -E` in the file `test`. Write an expression that will match the lines where the following appears:

    (α′) `a/4`

    (β′) `array[i]`
    Hint: Find a way to make the brackets protected from the shell, i.e. not a regular expression.

    (γ′) `be` as a separate word only (not contained in a larger word).
    Hint: Use the appropriate parameter to grep.

Activity with chain of command and regular expressions

12. Find the 5 commands you use most frequently in the terminal. Run the following commands in order. At each step, observe the output that the command gives you, and record your comments in the response file.

    Hint: The command `sed -e "s/regexp/replacement/flags"` replaces the regular expression `regexp` with the text `replacement`.

    In the expression `"s/^ //g"` given in step (γ′)

    - the `regexp` is the `space` character at the beginning of the line `'^ '`,
    - the `replacement` here is empty, i.e. nothing,
    - the `flag /g` says "do it everywhere".

    Overall, the command erases the spaces at the beginning of the line.

    (α′) `cat ~/.bash_history`

    (β′) `cat ~/.bash_history | tr "\|;" "\n"`

    (γ′) `cat ~/.bash_history | tr "\|;" "\n" | sed -e "s/^ //g"`

    (δ′) `cat ~/.bash_history | tr "\|;" "\n" | sed -e "s/^ //g" | cut -d " " -f 1`

    (ε′) `cat ~/.bash_history | tr "\|;" "\n" | sed -e "s/^ //g" | cut -d " " -f 1 | sort`

    (στ′) `cat ~/.bash_history | tr "\|;" "\n" | sed -e "s/^ //g" | cut -d " " -f 1 | sort | uniq -c`

    (ζ′) `cat ~/.bash_history | tr "\|;" "\n" | sed -e "s/^ //g" | cut -d " " -f 1 | sort | uniq -c | sort -n`

    (η′) `cat ~/.bash_history | tr "\|;" "\n" | sed -e "s/^ //g" | cut -d " " -f 1 | sort | uniq -c | sort -n | tail -5`

Submit your answers: https://forms.office.com/e/HNRXQJmv0n