# CS 1XC3 Lab 10 : Pipes, Filters and Regular Expressions

Apr. $3^{rd}$-$7^{th}$, 2023

## Contents



# 1 Introduction

In this lab, we're going to try our hand at creating some pretty complex commands that perform a variety of operations using glob and regex patterns.

Some of the problems in this lab are pretty challenging! It is our expectation that when you find something in this lab that you don't know how to do, you use your favourite search engine or the man pages to learn how a specific task might be accomplished.

Good luck!

## 1.1 Setup

For this lab, you will need to create a "L10" directory in your CS1XC3 master submission repository. You will need to download the compressed Archive "Lab10.tar.xz" from Avenue and decompress it somewhere other than your L10 folder. Many of the activities here will require you to perform searches and operations over this archive.

Some of the operations in this lab have a desctructive effect on this archive. It is advisable that you create a git archive (not a github archive!) that keeps track of the state of the files. That way, in order to return the archive to its original state, you only need to perform a hard reset.

# 2 Activity : Interactive Regular Expressions [8 points]

Let's get some practice with Regular Expressions!

1. In your "L10" directory, create a file "part1.txt".
2. Go to the following webpage and complete both the tutorial and the extra problems. Copy each of your answers into "part1.txt".
   - `https://regexone.com/lesson`
3. Commit your file to finish this activity.

# 3 Activity : Finding Files using Glob Patterns [15 points]

For this portion of the lab, you will be designing bash commands which perform the functions described below. You will need to use both glob patterns and pipes to perform these tasks.

Some of these commands will change the files you are designing commands for. It is **highly** recommended that you store the contents of the compressed archive file mentioned in the setup section in a git repo, so that you can use `git reset -hard` to undo your local changes quickly and easily. Note: please do not include the contents of the compressed archive in your submission repo.

While some of these tasks will have multiple valid approaches, all correct solutions take only one line. (For the purposes of this activity, we are considering bash commands which pipe together several individual commands to be one command).

Record your answers in "part2.txt", and commit it to your CS1XC3/L10 repo directory when finised.

1. Find all .tex files.
2. Find all .tex files excluding the file "template.tex".
3. Display the number of lines in each *.tex file.
4. Find all .tex, .pdf and .toc files.
5. Find the .pdf files from topics after topic 6.
6. Remove all gedit backup files (file names ending in ∼).
7. Remove all emacs backup files (file names beginning and ending with #).
8. Find only those image files (.png, .jpg, .bmp, .webp) whose file name (including extension) is exactly 10 characters long, and writes this list to a file named "goodFiles.txt".
9. Find all .log files and append them into one gigantic log file called "allTheLogs.log".
10. To standard output, print the 10th line of every *.aux file that has a capital letter C in the file name.
11. Add executable permission to every .sh file.
12. Append the first ten lines of every file name containing the work " and " (with spaces) to a file called "allheads.txt".
13. Copy every image file (.png, .jpg, .jpeg, and .webp) to a folder called image_backup.
14. Write a command which takes a file ("filenames.txt") containing file names and displays them in stdout.
15. Write a command which takes a file containing glob patterns called "allTheGlobs.txt", and finds all files matching those glob patterns.

# 4 Activity : Matching Text With `grep` [10 points]

In this section, you will be given a series of tasks, which you must design a one-line command to perform. Please record your answers in "part3.txt" in your CS1XC3/L10, and commit your work when finished. Again, these commands will operate over the files in the compressed archive provided on Avenue.

For these problems, you don't have to restrict the search to a specific file type unless otherwise specified.

1. Find every instance of `\includegraphics` in the *.tex files in the archive.
2. Find every `\begin{<tag name>}` tag in *.tex files, but exclude every tag with:
   - two sets of curly braces
     - i.e., exclude `\begin {<something>}{<something else>}`
   - A set of curly braces followed by square braces
     - i.e., exclude `\begin {<something>}[<something else>]`
3. Find every occurance of the words "arbitrary" "bitwidth" and "pointer" in files of any type.

4. For all .tex files with a capital "S" in the file name, Find every use of the word 'to be', in following tenses:

   - present
   - simple past
   - future
   - present perfect
   - past perfect
   - future perfect

   Conjugations of "to be" in English - `https://www.wordreference.com/conj/EnVerbs.aspx?v=be`

5. Using multi line matching, find `\begin{itemize}` immediately followed by `\item` on the next line.

6. Find all instances of the word "pointer" being followed by a word beginning with the letters R, S, T, L, N, and E, in either case. You must also account for instances of the word "pointer" enclosed in curly braces.

7. Find all words containing a double-vowel, begin and end with a space character.

8. Pattern match all C-style if statements.

```
if (<condition>) {
  ...
}
```

   Remember, this is a multi-line match!

9. Find the first line of all C functions. You can assume:

   - One space character separates each element, like so:

```
int main (void) {
```

   - You only need to account for return types of:
     - `int`
     - `void`
     - `float`
     - `char`

     However, you must account for any level of pointer (double, triple, etc).
   - You must account for any number of arguments.

10. Pattern match all arithmetic expressions of any size using only integers and the operators $+$, $-$, $/$, and $*$, assuming one space between each element.

# 5 Activity : Find and Replace Using `sed` [10 points]

In this section, you will be once again given a series of tasks, which you must design a one-line command to perform. Please record your answers in "part4.txt" in your CS1XC3/L10 repo directory, and commit your work when finished. Again, these commands will operate over the files in the compressed archive provided on Avenue.

1. In every .txt file, replace the word "backwards" with "forwards".

2. In every .tex file, find every graphic inserted using `\includegraphics[...]{...}` and delete it.

3. Find every \begin{itemize} and replace it for \begin{enumerate}

4. Include in superscript "[citation needed]" before each period.

5. Across all *.tex files, replace any word with the `-ing` suffix with the words "a gerund", but only if it occurs between the 100th and 300th line in the file.

6. Delete lines 15 through 18 in all *.log files.

7. Change all bold typeface to italics in all *.tex files.
8. Remove all vowels (including upper and lower case, but not including the letter "Y") from all *.tex and *.txt files.
9. Remove all comments in all *.tex files. In LaTeX, only single line comments are supported (in the same style as Python's # character). Everything after a % character not preceded by a \character is a comment.
10. Change all punctuation marks in *.tex and *.txt files into exclamation points!

# 6 In Closing...

Hopefully you now have an appreciation of the sheer power of the command line. The only reason these techniques work is because of the Unix standard of most files being stored in plain text. It is the universality of the Unix philosophy that enables such powerful techniques.

Congratulations! You are now a Linux expert. Now go make Linus Torvalds proud.