

Good Afternoon

CEG2350

Aiden Cox - CEG 2350 Lab Lead
Austin Kellough - CEG 2350 Lab Assistant

Quote of the week:

“The tree falls the way it leans. Be careful which way you lean”

How was Lab04?

Questions over anything up until now?
Scripting or link related? \$PATH?



Beginning Lab05

Lab Instructions: <https://pattonsgirl.github.io/CEG2350/Labs/Lab05/Instructions.html>

Lab Template:

<https://raw.githubusercontent.com/pattonsgirl/CEG2350/refs/heads/main/docs/Labs/Lab05/LabTemplate.md>

How are classes going?

Lets snag some files – Exam review

The screenshot shows a GitHub repository page for 'CEG2350 Midterm 1 Review'. The page has a header with a 'README' file icon and edit/clone buttons. The main content includes a section titled 'CEG2350 Midterm 1 Review' with a description of it being a comprehensive practice quiz covering essential Linux/Unix system administration concepts and commands. Below this is an 'Overview' section stating there are 54 questions covering topics like shells, file structures, SSH, and command-line tools. A question 'wget?' is highlighted in blue at the end of the list. A large blue link at the bottom provides the GitHub URL for the repository.

README

CEG2350 Midterm 1 Review

A comprehensive practice quiz covering essential Linux/Unix system administration concepts and commands.

Overview

This quiz contains 54 questions covering the following topics:

- Shells (bash, PowerShell, etc.)
- Files, directories, and OS structure
- SSH (Secure Shell) connections and configuration
- Bash scripting fundamentals
- File permissions and user management
- Text processing with grep, sed, and awk
- Regular expressions
- Git version control
- Practical command-line exercises

wget?

<https://github.com/aidenc17/CEG2350-Exam1-Review>

grep - Part 1

Part 1 - grep

grep is a handy command to find patterns in text. There are two flags that enable enhanced regular expressions: `-E` and `-P`. `-E` handles most things, but does not work with special regex letter the represent ranges `\w` and `\d` for example. To use these characters for the ranges they represent, use the `-P` flag instead of the `-E` flag.

```
grep [OPTIONS] PATTERN [FILE...]
```

In `access.log` you'll find dummy logs for users (client machines) accessing a web server. Each line contains:

- the client IP
- the access (request) timestamp
- the HTTP Request Method (GET or POST) & resource URL (page)
- the HTTP Status code

Your task is to use both `grep` and `wc` to parse the file for information and report on how many instances were found. Your searches with `grep` should add in enough patterning to reduce mistakes (inaccurate catches).

GREP(1) General Commands Manual GREP(1)

NAME String In A File Using REGEX grep, egrep, fgrep, rgrep - print lines matching a pattern

SYNOPSIS grep [OPTIONS] PATTERN [FILE...] with the following switches:
grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]

DESCRIPTION * Robin Hood
grep searches the named input FILES (or standard input if no files are named, or if a single hyphen-minus (-) is given as file name) for lines containing a match to the given PATTERN. By default, grep prints the matching lines.
Show White And The Seven Dwarfs
In addition, three variant programs egrep, fgrep and rgrep are available. egrep is the same as grep -E. fgrep is the same as grep -F. rgrep is the same as grep -r. Direct invocation as either egrep or fgrep is deprecated, but is provided to allow historical applications that rely on them to run unmodified.

Meta description of
Broken links

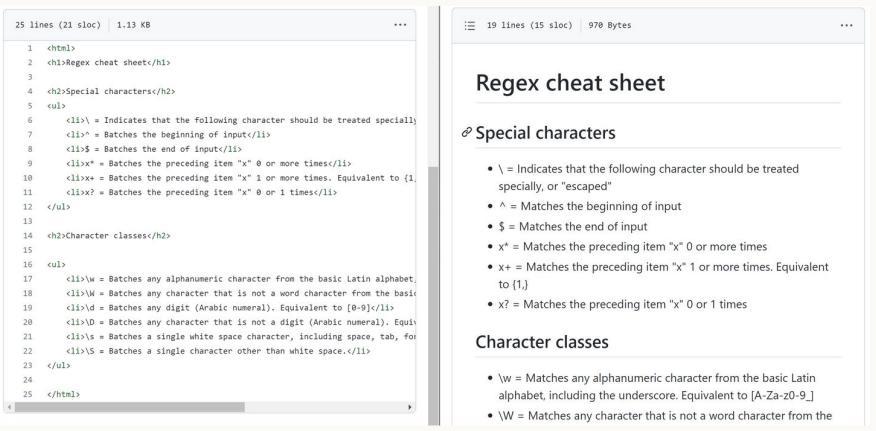
sed - Part 2

Part 2 - sed

sed is mostly commonly used as a search and replace command. In [sedfile.html](#) you'll find an html file. Your task is to change it into markdown using sed commands. Your converted file should be named `sedfile.md`.

- [Raw version of sedfile.html](#)

The following picture shows how `sedfile.html` (the original) looks compared to `sedfile.md` after being fixed to use markdown when the files are viewed in GitHub.



The image shows two GitHub code preview cards side-by-side. The left card, labeled '25 lines (21 sloc) 1.13 KB', displays the raw HTML content of `sedfile.html`. The right card, labeled '19 lines (15 sloc) 970 Bytes', displays the converted Markdown content of `sedfile.md`. Both cards show the same content, which is a 'Regex cheat sheet' document.

```
25 lines (21 sloc) 1.13 KB
...
1 <html>
2 <h1>Regex cheat sheet</h1>
3
4 <h2>Special characters</h2>
5 <ul>
6   <li>\ = Indicates that the following character should be treated specially</li>
7   <li>^ = Matches the beginning of input</li>
8   <li>$ = Matches the end of input</li>
9   <li>x* = Matches the preceding item "x" 0 or more times</li>
10  <li>x+ = Matches the preceding item "x" 1 or more times. Equivalent to {1,}</li>
11  <li>x? = Matches the preceding item "x" 0 or 1 times</li>
12 </ul>
13
14 <h2>Character classes</h2>
15
16 <ul>
17   <li>\w = Matches any alphanumeric character from the basic Latin alphabet; any underscore, and any digit from 0-9</li>
18   <li>\W = Matches any character that is not a word character from the basic Latin alphabet, including the underscore. Equivalent to [A-Za-z0-9_]</li>
19   <li>\d = Matches any digit (Arabic numeral). Equivalent to [0-9]</li>
20   <li>\D = Matches any character that is not a digit (Arabic numeral). Equivalent to [^0-9]</li>
21   <li>\s = Matches a single white space character, including space, tab, for new line</li>
22   <li>\S = Matches a single character other than white space.</li>
23 </ul>
24
25 </html>
```

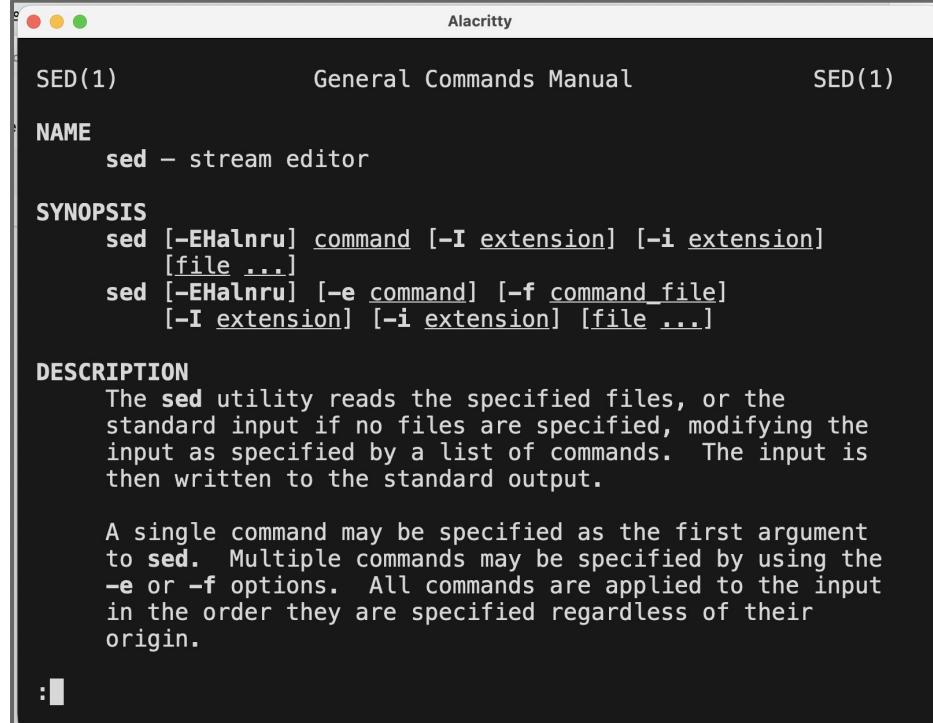
Regex cheat sheet

Special characters

- \ = Indicates that the following character should be treated specially, or 'escaped'
- ^ = Matches the beginning of input
- \$ = Matches the end of input
- x* = Matches the preceding item "x" 0 or more times
- x+ = Matches the preceding item "x" 1 or more times. Equivalent to {1,}
- x? = Matches the preceding item "x" 0 or 1 times

Character classes

- \w = Matches any alphanumeric character from the basic Latin alphabet, including the underscore. Equivalent to [A-Za-z0-9_]
- \W = Matches any character that is not a word character from the basic Latin alphabet, including the underscore. Equivalent to [^A-Za-z0-9_]



The image shows a terminal window titled 'Alacritty' displaying the 'General Commands Manual' for the 'SED(1)' command. The window title bar says 'Alacritty'. The main content area has three sections: 'NAME', 'SYNOPSIS', and 'DESCRIPTION'. The 'NAME' section contains the command name 'sed - stream editor'. The 'SYNOPSIS' section lists the command syntax: 'sed [-E|-H|-n|-r|-u] [command] [-I extension] [-i extension] [file ...]' and 'sed [-E|-H|-n|-r|-u] [-e command] [-f command_file] [-I extension] [-i extension] [file ...]'. The 'DESCRIPTION' section provides a detailed explanation of the utility's purpose and usage.

Alacritty

General Commands Manual SED(1)

NAME
`sed` – stream editor

SYNOPSIS

```
sed [-E|-H|-n|-r|-u] command [-I extension] [-i extension]
[file ...]
sed [-E|-H|-n|-r|-u] [-e command] [-f command_file]
[-I extension] [-i extension] [file ...]
```

DESCRIPTION

The `sed` utility reads the specified files, or the standard input if no files are specified, modifying the input as specified by a list of commands. The input is then written to the standard output.

A single command may be specified as the first argument to `sed`. Multiple commands may be specified by using the `-e` or `-f` options. All commands are applied to the input in the order they are specified regardless of their origin.

awk - Part 3

Part 3 - awk

awk is a full blown scripting language dedicated to text manipulation. See the resources for more examples, but the general format is:

```
awk 'program' input-file
```

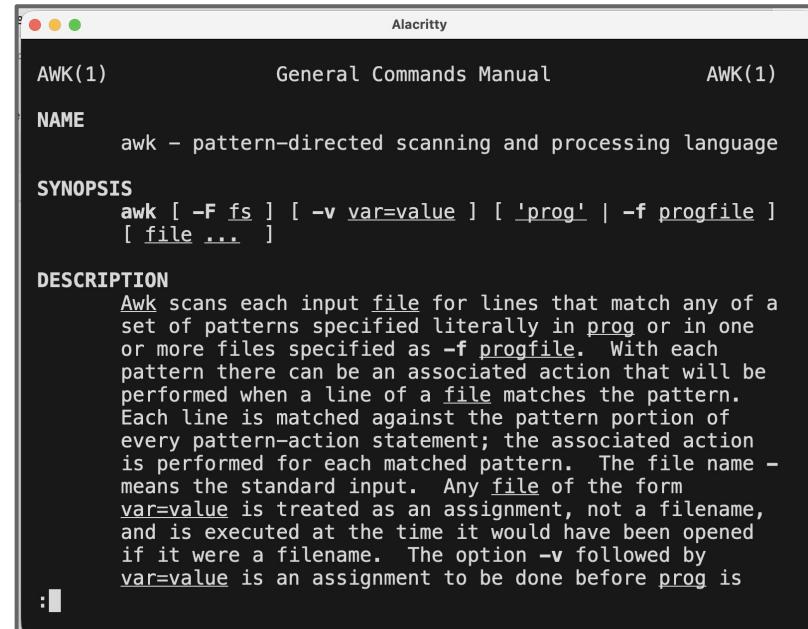
In [sales.txt](#) you'll find a file that contains sales records. Your task is to ask some queries and make replacements using awk.

- [Raw version of sales.txt](#)

Each lines in `sales.txt` contains the following fields:

- Date (YYYY-MM-DD)
- Product (String)
- Category (String)
- Quantity Sold (Integer)
- Unit Price (Float)
- Total Revenue (Quantity Sold × Unit Price)

Only write the command that performs the required task in the lab write up. DO NOT paste the results.



The screenshot shows a terminal window titled "Awk(1)" running in Alacritty. The window displays the "General Commands Manual" for Awk(1). The "NAME" section defines awk as a pattern-directed scanning and processing language. The "SYNOPSIS" section shows the command syntax: awk [-F fs] [-v var=value] ['prog' | -f progfile] [file ...]. The "DESCRIPTION" section provides a detailed explanation of how awk processes input files, matching patterns and performing actions, and how it handles assignments and standard input.

```
Awk(1)                               General Commands Manual                               AwK(1)

NAME
      awk - pattern-directed scanning and processing language

SYNOPSIS
      awk [ -F fs ] [ -v var=value ] [ 'prog' | -f progfile ]
          [ file ... ]

DESCRIPTION
      Awk scans each input file for lines that match any of a
      set of patterns specified literally in prog or in one
      or more files specified as -f progfile. With each
      pattern there can be an associated action that will be
      performed when a line of a file matches the pattern.
      Each line is matched against the pattern portion of
      every pattern-action statement; the associated action
      is performed for each matched pattern. The file name -
      means the standard input. Any file of the form
      var=value is treated as an assignment, not a filename,
      and is executed at the time it would have been opened
      if it were a filename. The option -v followed by
      var=value is an assignment to be done before prog is
```

Have a Good Weekend!

Don't hesitate to reach out and ask questions!

Quote of the week:

“The tree falls the way it leans. Be careful which way you lean”