Regular Expressions in Practice

Matthias Braun

Today's Plan

- Learn about a few Unix tools that make you look like a wizard

Today's Plan

- Learn about a few Unix tools that make you look like a wizard
- Use regexes with those tools

Today's Plan

- Learn about a few Unix tools that make you look like a wizard
- Use regexes with those tools
- Get your command-line skills to the next level

Get Unix

- We'll be using Unix command-line tools again

Get Unix

- We'll be using Unix command-line tools again
- If you're on Windows, you can use WSL, Cygwin, or Linux in the browser

Get Unix

- We'll be using Unix command-line tools again
- If you're on Windows, you can use WSL, Cygwin, or Linux in the browser
- Code for the command line looks like this:

```
printf 'Hi there! :-)'
```

Get Unix

- We'll be using Unix command-line tools again
- If you're on Windows, you can use WSL, Cygwin, or Linux in the browser
- Code for the command line looks like this:

```
printf 'Hi there! :-)'
```

- Paste the code into your command line to try it

Filtering a File

- You have a log file, `program.log`, with thousands of lines

Filtering a File

- You have a log file, `program.log`, with thousands of lines
- Lines in `program.log` look like this:

```
INFO: All systems normal
INFO: Still good. Current date is 2025-04-15
WARNING: Something strange happened...
INFO: Carrying on. Current date is 2025-04-16
WARNING: Possible data loss
WARNING: Something strange happened...
ERROR: Program is about to crash!
```

Filtering a File

- You have a log file, `program.log`, with thousands of lines
- Lines in `program.log` look like this:

```
INFO: All systems normal
INFO: Still good. Current date is 2025-04-15
WARNING: Something strange happened...
INFO: Carrying on. Current date is 2025-04-16
WARNING: Possible data loss
WARNING: Something strange happened...
ERROR: Program is about to crash!
```

- You want to see only warnings and errors

Filtering a File

- `grep` prints lines if a regex matches

Filtering a File

- `grep` prints lines if a regex matches
- Let's create a regex to match these sequences of characters:

  "WARNING"
  "ERROR"

Filtering a File

- `grep` prints lines if a regex matches
- Let's create a regex to match these sequences of characters:

  "WARNING"
  "ERROR"
- This should work:

```
grep -E 'WARNING|ERROR' program.log
```

Filtering a File

- `grep` prints lines if a regex matches
- Let's create a regex to match these sequences of characters:

  "WARNING"
  "ERROR"
- This should work:

```
grep -E 'WARNING|ERROR' program.log
```

- Now write those warnings and errors into a new file

Filtering a File

- `grep` prints lines if a regex matches
- Let's create a regex to match these sequences of characters:

  "WARNING"
  "ERROR"
- This should work:

```
grep -E 'WARNING|ERROR' program.log
```

- Now write those warnings and errors into a new file
- Redirect `grep`'s output with the `>` character:

```
grep -E 'WARNING|ERROR' program.log > warnings_and_errors.log
```

Reading a File

- Print the file contents with
  `cat warnings_and_errors.log`

Reading a File

- Print the file contents with
  ```
  cat warnings_and_errors.log
  ```
- Open it in a text editor like Vim:
  ```
  vim warnings_and_errors.log
  ```

Reading a File

- Print the file contents with
  `cat warnings_and_errors.log`
- Open it in a text editor like Vim:
  `vim warnings_and_errors.log`
  - Move the cursor up and down with k and j

Reading a File

- Print the file contents with
  `cat warnings_and_errors.log`
- Open it in a text editor like Vim:
  `vim warnings_and_errors.log`
  - Move the cursor up and down with k and j
  - If you've started writing into the file (this happens after you press i or a), return to normal mode by pressing Esc

Reading a File

- Print the file contents with
  `cat warnings_and_errors.log`
- Open it in a text editor like Vim:
  `vim warnings_and_errors.log`
  - Move the cursor up and down with k and j
  - If you've started writing into the file (this happens after you press i or a), return to normal mode by pressing Esc
  - Exit Vim with

Reading a File

- Print the file contents with
  ```
  cat warnings_and_errors.log
  ```
- Open it in a text editor like Vim:
  ```
  vim warnings_and_errors.log
  ```
  - Move the cursor up and down with k and j
  - If you've started writing into the file (this happens after you press i or a), return to normal mode by pressing Esc
  - Exit Vim with

    :x Enter (save and quit)

Reading a File

- Print the file contents with
  `cat warnings_and_errors.log`
- Open it in a text editor like Vim:
  `vim warnings_and_errors.log`
  - Move the cursor up and down with k and j
  - If you've started writing into the file (this happens after you press i or a), return to normal mode by pressing Esc
  - Exit Vim with

    `:x` Enter (save and quit)

    `:q!` Enter (quit without saving)

Reading a File

- Print the file contents with
  `cat warnings_and_errors.log`
- Open it in a text editor like Vim:
  `vim warnings_and_errors.log`
  - Move the cursor up and down with k and j
  - If you've started writing into the file (this happens after you press i or a), return to normal mode by pressing Esc
  - Exit Vim with
    `:x` Enter (save and quit)
    `:q!` Enter (quit without saving)
  - Run `vimtutor` in your command line to learn Vim, I highly recommend it

Downloading a File

The Hacker Way

- Imagine the log file is on another computer, a server

Downloading a File

The Hacker Way

- Imagine the log file is on another computer, a server
- Can we download a file using the command line? Easy:

```
curl -sL https://tinyurl.com/prg-log
```

Downloading a File

The Hacker Way

- Imagine the log file is on another computer, a server
- Can we download a file using the command line? Easy:

```
curl -sL https://tinyurl.com/prg-log
```

- `curl` gets the file at the given URL, prints it to the screen:

Downloading a File

The Hacker Way

- Imagine the log file is on another computer, a server
- Can we download a file using the command line? Easy:
  `curl -sL https://tinyurl.com/prg-log`
- `curl` gets the file at the given URL, prints it to the screen:
  - `-s` makes `curl` not show download progress

Downloading a File

The Hacker Way

- Imagine the log file is on another computer, a server
- Can we download a file using the command line? Easy:
  `curl -sL https://tinyurl.com/prg-log`
- `curl` gets the file at the given URL, prints it to the screen:
  - `-s` makes `curl` not show download progress
  - `-L` makes `curl` follow HTTP redirects

Downloading a File

The Hacker Way

- Imagine the log file is on another computer, a server
- Can we download a file using the command line? Easy:
  `curl -sL https://tinyurl.com/prg-log`
- `curl` gets the file at the given URL, prints it to the screen:
  - `-s` makes `curl` not show download progress
  - `-L` makes `curl` follow HTTP redirects
  - `curl` works also without the protocol part of the URL:
    `curl -sL tinyurl.com/prg-log`

Your Turn!

- Can you view the log file at the same URL https://tinyurl.com/prg-log with your web browser?

Your Turn!

- Can you view the log file at the same URL https://tinyurl.com/prg-log with your web browser?
- Use `curl` and output redirection `>` to save the file to disk

Your Turn!

- Can you view the log file at the same URL https://tinyurl.com/prg-log with your web browser?
- Use `curl` and output redirection `>` to save the file to disk
- Use `curl` to read the news at `https://orf.at`

Your Turn!

- Can you view the log file at the same URL https://tinyurl.com/prg-log with your web browser?
- Use `curl` and output redirection `>` to save the file to disk
- Use `curl` to read the news at `https://orf.at`
- Read the content of a website you visit often with `curl`

Piping

Makes the output of one program the input to another program: |

- We can download a file and save it to disk with
  `curl -sL tinyurl.com/prg-log > program.log`
  and filter it with
  `grep -E 'WARNING|ERROR' program.log`

Piping

Makes the output of one program the input to another program: |

- We can download a file and save it to disk with
  `curl -sL tinyurl.com/prg-log > program.log`
  and filter it with
  `grep -E 'WARNING|ERROR' program.log`
- We can download and filter it with a combined command:
  `curl -sL tinyurl.com/prg-log | grep -E 'WARNING|ERROR'`

Piping

Makes the output of one program the input to another program: |

- We can download a file and save it to disk with
  `curl -sL tinyurl.com/prg-log > program.log`
  and filter it with
  `grep -E 'WARNING|ERROR' program.log`
- We can download and filter it with a combined command:
  `curl -sL tinyurl.com/prg-log | grep -E 'WARNING|ERROR'`
- Now, `grep` filters the output of `curl` instead of a file on disk

Piping

Makes the output of one program the input to another program: `|`

- We can download a file and save it to disk with
  `curl -sL tinyurl.com/prg-log > program.log`
  and filter it with
  `grep -E 'WARNING|ERROR' program.log`
- We can download and filter it with a combined command:
  `curl -sL tinyurl.com/prg-log | grep -E 'WARNING|ERROR'`
- Now, `grep` filters the output of `curl` instead of a file on disk
- Using a `|` pipe, `curl`'s output becomes `grep`'s input

More Piping

Makes the output of one program the input to another program: |

- Use multiple pipes to build a sequence of commands:

More Piping

Makes the output of one program the input to another program: |

- Use multiple pipes to build a sequence of commands:
  - Print some numbers: `printf '2\n4\n3\n3\n1\n4\n'`

More Piping

Makes the output of one program the input to another program: |

- Use multiple pipes to build a sequence of commands:
  - Print some numbers: `printf '2\n4\n3\n3\n1\n4\n'`
  - Sort the numbers: `printf '2\n4\n3\n3\n1\n4\n' | sort`

More Piping

Makes the output of one program the input to another program: |

- Use multiple pipes to build a sequence of commands:
  - Print some numbers: `printf '2\n4\n3\n3\n1\n4\n'`
  - Sort the numbers: `printf '2\n4\n3\n3\n1\n4\n' | sort`
  - Get rid of duplicates:
    `printf '2\n4\n3\n3\n1\n4\n' | sort | uniq`

More Piping

Makes the output of one program the input to another program: |

- Use multiple pipes to build a sequence of commands:
  - Print some numbers: `printf '2\n4\n3\n3\n1\n4\n'`
  - Sort the numbers: `printf '2\n4\n3\n3\n1\n4\n' | sort`
  - Get rid of duplicates:
    `printf '2\n4\n3\n3\n1\n4\n' | sort | uniq`
- What happens in the last command:

More Piping

Makes the output of one program the input to another program: |

- Use multiple pipes to build a sequence of commands:
  - Print some numbers: `printf '2\n4\n3\n3\n1\n4\n'`
  - Sort the numbers: `printf '2\n4\n3\n3\n1\n4\n' | sort`
  - Get rid of duplicates:
    `printf '2\n4\n3\n3\n1\n4\n' | sort | uniq`
- What happens in the last command:
  - First, `printf` creates lines with a single number on them.

    The pipe `|` sends those lines to `sort`

More Piping

Makes the output of one program the input to another program: |

- Use multiple pipes to build a sequence of commands:
  - Print some numbers: `printf '2\n4\n3\n3\n1\n4\n'`
  - Sort the numbers: `printf '2\n4\n3\n3\n1\n4\n' | sort`
  - Get rid of duplicates:
    `printf '2\n4\n3\n3\n1\n4\n' | sort | uniq`
- What happens in the last command:
  - First, `printf` creates lines with a single number on them.

    The pipe `|` sends those lines to `sort`
  - Then, `sort` sorts the lines and `|` sends them to `uniq`

More Piping

Makes the output of one program the input to another program: |

- Use multiple pipes to build a sequence of commands:
  - Print some numbers: `printf '2\n4\n3\n3\n1\n4\n'`
  - Sort the numbers: `printf '2\n4\n3\n3\n1\n4\n' | sort`
  - Get rid of duplicates:
    `printf '2\n4\n3\n3\n1\n4\n' | sort | uniq`
- What happens in the last command:
  - First, `printf` creates lines with a single number on them.

    The pipe `|` sends those lines to `sort`
  - Then, `sort` sorts the lines and `|` sends them to `uniq`
  - Then, `uniq` removes a line if it is identical to the line before

More Piping

Makes the output of one program the input to another program: |

- Use multiple pipes to build a sequence of commands:
  - Print some numbers: `printf '2\n4\n3\n3\n1\n4\n'`
  - Sort the numbers: `printf '2\n4\n3\n3\n1\n4\n' | sort`
  - Get rid of duplicates:
    `printf '2\n4\n3\n3\n1\n4\n' | sort | uniq`
- What happens in the last command:
  - First, `printf` creates lines with a single number on them.

    The pipe `|` sends those lines to `sort`
  - Then, `sort` sorts the lines and `|` sends them to `uniq`
  - Then, `uniq` removes a line if it is identical to the line before
  - Finally, `uniq` outputs the remaining lines

Filtering a Remote File

Piping Intensifies

- Enter this as a single line:

```
curl -sL tinyurl.com/prg-log
  | grep -E 'WARNING|ERROR'
  | sort
  | uniq
  > output.log
```

Filtering a Remote File

Piping Intensifies

- Enter this as a single line:

```
curl -sL tinyurl.com/prg-log
  | grep -E 'WARNING|ERROR'
  | sort
  | uniq
  > output.log
```

- The output contains the sorted warnings and errors without duplicates

Your Turn!

Log File

- What happens if you remove `| uniq` from the previous command?

Your Turn!

Log File

- What happens if you remove `| uniq` from the previous command?
- Modify the previous command to see `INFO` and `ERROR` messages

Your Turn!

Log File

- What happens if you remove `| uniq` from the previous command?
- Modify the previous command to see `INFO` and `ERROR` messages
- Make `grep` print line numbers at the start of the line. Use `grep --help` to remind yourself how to do it

Your Turn!

Log File

- What happens if you remove `| uniq` from the previous command?
- Modify the previous command to see `INFO` and `ERROR` messages
- Make `grep` print line numbers at the start of the line. Use `grep --help` to remind yourself how to do it
- What's the difference between `>` and `>>`?

Your Turn!

Log File

- What happens if you remove `| uniq` from the previous command?
- Modify the previous command to see `INFO` and `ERROR` messages
- Make `grep` print line numbers at the start of the line. Use `grep --help` to remind yourself how to do it
- What's the difference between `>` and `>>`?
- Create a regex that matches dates in the format of the log file:
  - 2025-04-15
  - 2025-04-16
  - 1999-12-14

  Use this regex with `grep` to print the two lines in the log with dates. Note that your regex shouldn't only match specific dates but *all* dates of this format. The format is **the** correct way of writing dates.

The dot

Match any character

- The dot `.` matches any character, except newline characters `\r` and `\n`

The dot

Match any character

- The dot `.` matches any character, except newline characters `\r` and `\n`
- For example, `.at` matches “cat”, “Bat”, and “-at”

The dot

Match any character

- The dot `.` matches any character, except newline characters `\r` and `\n`
- For example, `.at` matches “cat”, “Bat”, and “-at”
- We want to match lines with “one” and then “two” in them:

```
one → two (match this)
zero, one, two, three (match this)
one (don't match)
two (don't match)
two one (don't match)
```

The dot

Match any character

- The dot `.` matches any character, except newline characters `\r` and `\n`
- For example, `.at` matches “cat”, “Bat”, and “-at”
- We want to match lines with “one” and then “two” in them:

```
one → two (match this)
zero, one, two, three (match this)
one (don't match)
two (don't match)
two one (don't match)
```

- The regex is `one.+two`: First “one”, then any characters, and finally “two”

Your Turn!

Romeo and Juliet

- Lots of text: tinyurl.com/rom-jul

Your Turn!

Romeo and Juliet

- Lots of text: tinyurl.com/rom-jul
- Write a `grep` command that prints
  `2910:O Romeo, Romeo! wherefore art thou Romeo?`

Your Turn!

Romeo and Juliet

- Lots of text: tinyurl.com/rom-jul
- Write a `grep` command that prints
  `2910:O Romeo, Romeo! wherefore art thou Romeo?`
- Write a command that counts how many lines with "O Romeo" there are in the text. Hint: use `grep --help` or `wc --help`

Your Turn!

Romeo and Juliet

- Lots of text: tinyurl.com/rom-jul
- Write a `grep` command that prints
  `2910:O Romeo, Romeo! wherefore art thou Romeo?`
- Write a command that counts how many lines with "O Romeo" there are in the text. Hint: use `grep --help` or `wc --help`
- There are four lines that contain **both** "love" and "hate". For example:

Your Turn!

Romeo and Juliet

- Lots of text: tinyurl.com/rom-jul
- Write a `grep` command that prints
  `2910:O Romeo, Romeo! wherefore art thou Romeo?`
- Write a command that counts how many lines with "O Romeo" there are in the text. Hint: use `grep --help` or `wc --help`
- There are four lines that contain **both** "love" and "hate". For example:
  - "Here's much to do with hate, but more with love."

Your Turn!

Romeo and Juliet

- Lots of text: tinyurl.com/rom-jul
- Write a `grep` command that prints
  `2910:O Romeo, Romeo! wherefore art thou Romeo?`
- Write a command that counts how many lines with "O Romeo" there are in the text. Hint: use `grep --help` or `wc --help`
- There are four lines that contain **both** "love" and "hate". For example:
  - "Here's much to do with hate, but more with love."
  - "My only love sprung from my only hate!"

Your Turn!

Romeo and Juliet

- Lots of text: tinyurl.com/rom-jul
- Write a `grep` command that prints
  `2910:O Romeo, Romeo! wherefore art thou Romeo?`
- Write a command that counts how many lines with "O Romeo" there are in the text. Hint: use `grep --help` or `wc --help`
- There are four lines that contain **both** "love" and "hate". For example:
  - "Here's much to do with hate, but more with love."
  - "My only love sprung from my only hate!"
- What's the regex to find all four lines containing both "love" and "hate", in any order?

Search and Replace

Using sed

- sed searches text and replaces matched text

Search and Replace

Using `sed`

- `sed` searches text and replaces matched text
- For example, `printf 'He said no' | sed 's/no/yes/'` prints: `He said yes`

Search and Replace

Using `sed`

- `sed` searches text and replaces matched text
- For example, `printf 'He said no' | sed 's/no/yes/'` prints: `He said yes`
- Remember: the pipe `|` sends the output of `printf` to `sed`

Search and Replace

Using `sed`

- `sed` searches text and replaces matched text
- For example, `printf 'He said no' | sed 's/no/yes/'` prints: `He said yes`
- Remember: the pipe `|` sends the output of `printf` to `sed`
- `sed`'s syntax: `s/regex/replacement/`

Search and Replace

Using `sed`

- `sed` searches text and replaces matched text
- For example, `printf 'He said no' | sed 's/no/yes/'` prints: `He said yes`
- Remember: the pipe `|` sends the output of `printf` to `sed`
- `sed`'s syntax: `s/regex/replacement/`
  - the `s` stands for "substitute"

Search and Replace

Using `sed`

- `sed` searches text and replaces matched text
- For example, `printf 'He said no' | sed 's/no/yes/'` prints: `He said yes`
- Remember: the pipe `|` sends the output of `printf` to `sed`
- `sed`'s syntax: `s/regex/replacement/`
  - the `s` stands for "substitute"
  - the last `/` has to be there

Search and Replace

- `printf 'He said no no no' | sed 's/no/yes/'` prints:
  `He said yes no no`

Search and Replace

- `printf 'He said no no no' | sed 's/no/yes/'` prints:
  `He said yes no no`
- Keep replacing after the first match with the **g**lobal flag:
  `printf 'He said no no no' | sed 's/no/yes/g'`
  prints: `He said yes yes yes`

Search and Replace

Tabs and Spaces

- File with a mix of tabs and spaces: tinyurl.com/tabs-spaces

Search and Replace

Tabs and Spaces

- File with a mix of tabs and spaces: tinyurl.com/tabs-spaces
- On Linux, match a tab character with `\t`

Search and Replace

Tabs and Spaces

- File with a mix of tabs and spaces: tinyurl.com/tabs-spaces
- On Linux, match a tab character with `\t`
- For macOS' version of `sed` you can't use `\t`, instead enter a tab character in your terminal with ctrl + v, then →⇥

Search and Replace

Tabs and Spaces

- File with a mix of tabs and spaces: tinyurl.com/tabs-spaces
- On Linux, match a tab character with `\t`
- For macOS' version of `sed` you can't use `\t`, instead enter a tab character in your terminal with ctrl + v, then →⇥
- How to spot the difference between tabs and spaces? It's both whitespace

Search and Replace

Tabs and Spaces

- File with a mix of tabs and spaces: tinyurl.com/tabs-spaces
- On Linux, match a tab character with `\t`
- For macOS' version of `sed` you can't use `\t`, instead enter a tab character in your terminal with ctrl + v, then →⇥
- How to spot the difference between tabs and spaces? It's both whitespace
- Find tabs by replacing them with visible characters, like "TAB" (use a single line for the command):

```
curl -sL tinyurl.com/tabs-spaces
  | sed 's/\t/TAB/g'
```

Search and Replace

The Result

```
class TabsAndSpaces {

TABpublic static void main (String[] args){

    String text = "Tabs and spaces don't mix";
TABTABSystem.out.println(text);
  }
}
```

Your Turn!

Tabs and Spaces

- Change the previous command to replace each tab character with two spaces

Your Turn!

Tabs and Spaces

- Change the previous command to replace each tab character with two spaces
- Write the resulting text to a file `SpacesOnly.java`

Your Turn!

Tabs and Spaces

- Change the previous command to replace each tab character with two spaces
- Write the resulting text to a file `SpacesOnly.java`
- Do the opposite now: Replace two spaces with a tab and save the result as `TabsOnly.java`:

Your Turn!

Tabs and Spaces

- Change the previous command to replace each tab character with two spaces
- Write the resulting text to a file `SpacesOnly.java`
- Do the opposite now: Replace two spaces with a tab and save the result as `TabsOnly.java`:
  - `TabsOnly.java` should be identical to tinyurl.com/tabs-only

Your Turn!

Tabs and Spaces

- Change the previous command to replace each tab character with two spaces
- Write the resulting text to a file `SpacesOnly.java`
- Do the opposite now: Replace two spaces with a tab and save the result as `TabsOnly.java`:
  - `TabsOnly.java` should be identical to tinyurl.com/tabs-only
  - If you get no output from

```
diff TabsOnly.java <(curl -sL tinyurl.com/tabs-only)
```

    it means there are no differences between your local file and the remote file → Your solution is correct

Process Substitution

Advanced Bash Feature

```
diff TabsOnly.java <(curl -sL tinyurl.com/tabs-only)
```

Process Substitution

Advanced Bash Feature

```
diff TabsOnly.java <(curl -sL tinyurl.com/tabs-only)
```

`diff` expects two local files for comparing. The `<(curl ...)` part acts like a temporary file that contains the result of `curl`. This is called process substitution and is helpful when you want to use the output of a command as a file, without creating a temporary file

Your Turn!

- Change all occurrences of “gray” and “grey” in tinyurl.com/grey-txt to “yellow”. Save the new text to `yellow.txt`

Your Turn!

- Change all occurrences of "gray" and "grey" in tinyurl.com/grey-txt to "yellow". Save the new text to `yellow.txt`
- Hint: If you want to use grouping `()` or choice `|` in `sed`, you need to use the `-E` flag for extended regexes: `sed -E`

Your Turn!

- Change all occurrences of "gray" and "grey" in tinyurl.com/grey-txt to "yellow". Save the new text to `yellow.txt`
- Hint: If you want to use grouping `()` or choice `|` in `sed`, you need to use the `-E` flag for extended regexes: `sed -E`
- You know you got it right if `diff yellow.txt <(curl -sL tinyurl.com/yellow-txt)` prints nothing

Your Turn!

- Change all occurrences of "gray" and "grey" in tinyurl.com/grey-txt to "yellow". Save the new text to `yellow.txt`
- Hint: If you want to use grouping `()` or choice `|` in `sed`, you need to use the `-E` flag for extended regexes: `sed -E`
- You know you got it right if `diff yellow.txt <(curl -sL tinyurl.com/yellow-txt)` prints nothing
- How many challenges can you complete on cmdchallenge.com? Send me a screenshot

Your Turn!

- Change all occurrences of "gray" and "grey" in tinyurl.com/grey-txt to "yellow". Save the new text to `yellow.txt`
- Hint: If you want to use grouping `()` or choice `|` in `sed`, you need to use the `-E` flag for extended regexes: `sed -E`
- You know you got it right if `diff yellow.txt <(curl -sL tinyurl.com/yellow-txt)` prints nothing
- How many challenges can you complete on cmdchallenge.com? Send me a screenshot
- Optional: Become a hacker on OverTheWire