Regular Expressions

EECS 348 Lab 2 — 2/6/2025

Harlan Williams

Electrical Engineering and Computer Science
University of Kansas

Useful command-line tools

- `cat <file>`—outputs whatever its input is to the terminal
- `wc -l`—prints the number of lines in the input
- `grep -E <regex>`—searches input for occurrences of a regular expression
  - print lines that begin with a function: `grep -E "def" program.py`
- `sed -e s/<regex>/string/g`—replaces strings which match a regex with another string.
  - replace “hello” with “goodbye”: `echo "hello" | sed -e s/hello/goodbye/g`
- `awk`—succinct language for writing programs that use regular expressions
  - filter lines with words that have two consecutive o’s: `echo "hello\ngoodbye" | awk /oo/`

Building and running scripts

Most command-line tools can take input either from a file, or from other commands as part of a *pipeline*, commands separated by `|` characters.

For example, `cat <file>` just prints a file to the terminal, but `cat <file> | sort` will use the output of `cat` as the input to `sort`.

Sequences of commands can be saved in scripts (`.sh` files). To run these, you will need to make them executable with the command `chmod +x script.sh`. From there, type `./script.sh`

Regular expressions recap

- `hello` matches exactly the string “hello”.
- Match (or don’t match) characters in a set: `[abc]` matches any of a, b, or c; `[^abc]` matches any character other than a, b, or c.
- Match something
  - zero or one times: “?”
  - zero or more times: “*”
  - one or more times: “+”
  - between `x` and `y` times: “`{x,y}`”
- Either regular expression r_1 *or* r_2: “$r_1 \mid r_2$”
- Escape metacharacters with a backslash: `\(` matches `(`
- `\d` matches any digit, `\w` matches any word character
- `^` matches the beginning of a line, `$` matches the end of a line

Common regular expression usage

- Search through text, *e.g.* searching through a large codebase for class definitions or function definitions

  `grep -E "def [A-Za-z0-9_]+\(\):` matches python functions that take zero arguments

- User input validation, *e.g.* making sure the user only gives a valid looking email address in an input form

  `email.match(/[A-Za-z0-9]+@[A-Za-z0-9]+\.[a-z]{2,3}/)` ensures the variable `email` looks like a valid email address

Lost? See...

- `man <cmd>` for the manual page of a specific command
- Add `-E` to `grep` if your regular expression makes sense but seems not to work—this forces `grep` to use a newer regular expression syntax
- https://www.regular-expressions.info/ to read more on how regular expressions work
- https://regex101.com/ for an interactive regex tester