Extra Topics - sed and awk

OPS102 Week 6 Class 2

John Sellens based on ULI101 notes

July 2, 2024

Seneca Polytechnic

Outline

Remember

Remember – What We Like

- We like text files
- We like UNIX/Linux pipes
- We like filter commands
- We like the UNIX Philosophy
- And so today: `sed` and `awk`

Heads Up – Regular Expressions

- Regular expressions – powerful patterns
- More in week 12.
- But part of the power of `sed` and `awk` comes from regular expressions.

`sed` – the stream editor

sed – the stream editor

- We all know what a regular text editor is.
  - An interactive tool for changing text files.
- We all know what a UNIX/Linux filter is.
  - Reads input, processes it, provides output.
  - Handy for use in pipeline commands.
- sed, the stream editor is a filter that edits files as they pass by.
  - Relatively simple edits.
  - But very powerful

sed Basics

- `sed` takes one or more sets of instructions.
- Instructions are a line selector and an action.
  - May be separated by whitespace for readability.
  - The `"-e"` flag precedes instruction sets, optional if just one instruction set
- Considers each input line in turn.
- If line selector matches, then do the action.
- Output the (possibly) changed line – unless `"-n"` (no output) used.

```
sed [-n] -e 'selector action' filename
```

sed Selectors

Line selectors can be:

- A line number e.g. `sed -e '5d' myfile`
  - A dollar sign $ selects the last line
- A line range e.g. `sed -e '1,5d' myfile`
- A regular expression e.g. `sed -e '/cat/d' myfile`
- A missing selector means select all lines
- An exclamation mark after the selector, inverts the selection
  - e.g. `sed -e '/cat/!d' myfile`

sed Actions

Line actions include (there are many):

- p – print selected lines (most often used with the `"-n"` option)
- d – delete selected lines from output and further processing
- q – quit processing after the selected line
- s – substitute pattern for replacement
    - `sed -e '1,5 s/cat/dog/' myfile`

awk – The Filter Power Tool

awk – The Filter Power Tool

- awk takes the sed idea of selector/action to new heights.
- Has control statements – if/then/else, while, for, switch
- Has variables
- Is “Turing-complete” i.e. a complete programming language
- Can be used for large(-ish) programs
- Named for its authors: Alfred Aho, Peter Weinberger, and Brian Kernighan
- No default output – need to explicitly print any lines
- More at: https://en.wikipedia.org/wiki/AWK

awk Basics

```
awk [-F field_separator] 'selector action' myfile
```

- Reads each input line in turn
- If selector matches, action is performed
- Like `sed` but more powerful
- The *action* can be multiple, semi-colon separated, statements
- `awk` breaks each line into tokens based on the field separator setting (`"-F"` option, default whitespace)

awk Selectors

Line selectors can be:

- A line number e.g. `awk 'NR==5 { print; }' myfile`
- A line range e.g. `awk -e 'NR>4 && NR<10 { print }' myfile`
- A regular expression match e.g. `awk -e '/cat/' myfile`
- A regular expression field match e.g. `awk -e '$2 ~ /cat/' myfile`
- A regular expression negative match e.g. `awk -e '$2 !~ /cat/' myfile`
- Variables like $0 entire line, $2 second input token, NR record number, NF number of fields, etc.

Larger awk

- Special BEGIN and END selectors – action runs before first line, or after last line
  - Handy for initialization, printing introductions or summaries
- You can put an `awk` program in a file (with `"-f"` option)
  - You can with `sed` as well, but less readable

Summary

Summary

- We just scratched the surface of what `sed` and `awk` can do
- Lots more functionality – see the man pages
- Very handy tools to be familiar with