Table of Contents

Example: checking who has logged into a server

```
$ ssh myserver 'journalctl | grep sshd | grep "Disconnected from"' >
    ssh.log
$ less ssh.log
```

ssh into myserver. Query the systemd journal, search it for ssh related entries, and then search those for any containing the phrase "Disconnected from". Output it as a file ssh.log so that we don't have to keep connected to the server.
Then open the log with pagination using `less`

`sed`

- `sed`: stream editor, builds on top of old `ed` editor
  - give short commands for how to modify file, rather than directly manipulating contents
  - `s`: substitution; written in form `s/REGEX/SUBSTITUTION`
    - `REGEX`: regular expression to search for
    - `SUBSTITUTION`: the text you want to substitute matching text with

```
ssh myserver journalctl`
  | grep sshd
  | grep "Disconnected from"
  | sed 's/.*Disconnected from //'
```

Regular Expressions

- regexr
- used to match text to a specified pattern
- regular expressions are usually surrounded by `/`
- most ASCII characters have normally meanings, but some are special matching characters, with meaning varying from system to system

Typically:

- `.`: any single character, except newline
- `*`: 0+ of preceding match
- `+`: 1+ of preceding match
- `[abc]`: any one character of `a`, `b`, and `c`
- `(RX1|RX2)`: matches `RX1` or `RX2`
- `^`: start of the line
- `$`: end of the line

NB `sed` requires you to escape special characters with `\` for them to match. Typically the inverse is true.

- capture groups: any text surrounded by parentheses is stored in a numbered capture group, available as `\1`, `\2`, ...

```
| sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*) [^
    ]+ port [0-9]+( \[preauth\])?$/\2/'
```

This regex matches any number of any characters, followed by “Disconnected from”, followed by an optional “invalid” or “authenticating”, followed by “user”, following by any number of characters, followed by 1 or more characters other than space, followed by ” port “, followed by 1 or more digits, with an optional suffix of”[preauth]”, and then the end of the line. Keep the second capture group containing the username

Email address matching: harder than it seems

- ~99% of emails matched
- 99.99% of emails matched

Now sort by usernames, and collapse to distinct values, with a count:

```
ssh myserver journalctl
 | grep sshd
```

```
  | grep "Disconnected from"
  | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*)
    [^ ]+ port [0-9]+( \[preauth\])?$/\2/'
  | sort | uniq -c
  | sort -nk1,1  # -n: sort in numeric order;
                 # -k1,1: sort by first whitespace-separated column only
  | tail -n10    # take the 10 most common ones
  | awk '{print $2}' # awk is another editor.  see below
  | paste -sd,       # combines lines (-s) by delimiter ","
```

awk

- `awk` is a programming language;
- very good at processing text streams
- programs comprise an optional pattern plus a block specifying behaviour for matches on a given line. The default pattern matches all lines.
- Inside the block:
  - `$0`: entire line
  - `$1`-`$n`: n-th field of the line, when separated by the `awk` field (default: whitespace, specify alternative character with -F)
- `{print $2}` then means print the second field of each line (the username)
- see class notes for more detail

Data analysis

```
| paste -sd+ # add numbers on each line together
| bc -l
```

```
echo "2*($(data | paste -sd+)" | bc -l
```

Stats: `st` or R

```
ssh myserver journalctl
  | grep sshd
  | grep "Disconnected from"
  | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*)
    [^ ]+ port [0-9]+( \[preauth\])?$/\2/'
  | sort | uniq -c
  | awk '{print $1}' | R --slave -e 'x <- scan(file="stdin", quiet=TRUE)
    ; summary(x)'
```

- basic plotting: `gnuplot`

```
ssh myserver journalctl
  | grep sshd
  | grep "Disconnected from"
  | sed -E 's/.*Disconnected from (invalid |authenticating )?user (.*)
      [^ ]+ port [0-9]+( \[preauth\])?$/\2/'
  | sort | uniq -c
  | sort -nk1,1 | tail -n10
  | gnuplot -p -e 'set boxwidth 0.5; plot "-" using 1:xtic(2) with boxes
      '
```

- binary data: these tools also work for binary data.
  e.g. capture image with ffmpeg, convert to grayscale, compress it, send it to a remote machine, decompress it, make a copy, display it:

```
ffmpeg -loglevel panic -i /dev/video0 -frames 1 -f image2 -
  | convert - -colorspace gray -
  | gzip
  | ssh mymachine 'gzip -d | tee copy.jpg | env DISPLAY=:0 feh -'
```

Exercises

1. Take this short interactive regex tutorial.
2. Find the number of words (in `/usr/share/dict/words`) that contain at least three `a`s and don't have a `'s` ending. What are the three most common last two letters of those words? `sed` 's `y` command, or the `tr` program, may help you with case insensitivity. How many of those two-letter combinations are there? And for a challenge: which combinations do not occur?
3. To do in-place substitution it is quite tempting to do something like `sed s/REGEX/SUBSTITUTION/ input.txt > input.txt`. However this is a bad idea, why? Is this particular to `sed`? Use `man sed` to find out how to accomplish this.
4. Find your average, median, and max system boot time over the last ten boots. Use `journalctl` on Linux and `log show` on macOS, and look for log timestamps near the beginning and end of each boot. On Linux, they may look something like:

   ```
   Logs begin at ...
   ```

   and

   ```
   systemd[577]: Startup finished in ...
   ```

   On macOS, look for:

```
=== system boot:
```

and

```
Previous shutdown cause: 5
```

5. Look for boot messages that are *not* shared between your past three reboots (see `journalctl` 's `-b` flag). Break this task down into multiple steps. First, find a way to get just the logs from the past three boots. There may be an applicable flag on the tool you use to extract the boot logs, or you can use `sed '0,/STRING/d'` to remove all lines previous to one that matches `STRING`. Next, remove any parts of the line that *always* varies (like the timestamp). Then, de-duplicate the input lines and keep a count of each one (`uniq` is your friend). And finally, eliminate any line whose count is 3 (since it *was* shared among all the boots).
6. Find an online data set like this one, this one. or maybe one from here. Fetch it using `curl` and extract out just two columns of numerical data. If you're fetching HTML data, `pup` might be helpful. For JSON data, try `jq`. Find the min and max of one column in a single command, and the sum of the difference between the two columns in another.

Solutions

1. completed
2. Number of words with ≥ 3 a's not finishing in 's

```
grep -E "\w*[aA]\w*a\w*a\w*[^(\'s)]$" words \
| wc -w # word count, with output words flag
# 435
```

Equivalently

```
grep -c -E "\w*[aA]\w*a\w*a\w*[^(\'s)]$" words
# 435
```

Three most common last two letters

```
grep -E "^\w*[aA]\w*a\w*a\w*[^(\'s)]$" words
| sed -E 's/^\w*(\w{3})$/\1/'
| sort
| uniq -c
| sort -rnk1,1
| head -n3
# 53 ian
# 31 lly
# 20 ion
```

1. As `sed` operates on file streams, you would be reading and writing to the same file at the same time. The os may not allow this, and if it did things could end up corrupted. There is an inplace flag `-i` for `sed` Which allows you to do this.
2. My `journalctl` doesn't have any entries (WSL). ssh into one of my raspberry pi's and download it's journalctl It doesn't appear to have any startup info. There is literally nothing interesting in the file…

3. <– todo –>
4.

```
curl https://ucr.fbi.gov/crime-in-the-u.s/2016/crime-in-the-u.s.-2016/topic-pages/tables/table-1 > crime_rate.html
cat crime_rate.html | pup 'table .group0,.group3 text{}' | grep -E "[^\n]" | grep -oE "([0-9]{4})" > year
cat crime_rate.html | pup 'table .group0,.group3 text{}' | grep -E "[^\n]" | grep -oE "([0-9]{3}\.[0-9])" > violent_crime_rate
paste -d " " year violent_crime_rate > crime_rate
```

Note this is an instance where you could use `tee` to pipe the common output of the first grep to each second grep, and then use this as an input to the paste.

year	violent_crime_rate
1997	611.0
1998	567.6
1999	523.0
2000	506.5
2001	504.5
2002	494.4
2003	475.8
2004	463.2
2005	469.0
2006	479.3
2007	471.8
2008	458.6
2009	431.9
2010	404.5

year	violent_crime_rate
2011	387.1
2012	387.8
2013	369.1
2014	361.6
2015	373.7
2016	386.3

Subtract column 2 from column 1

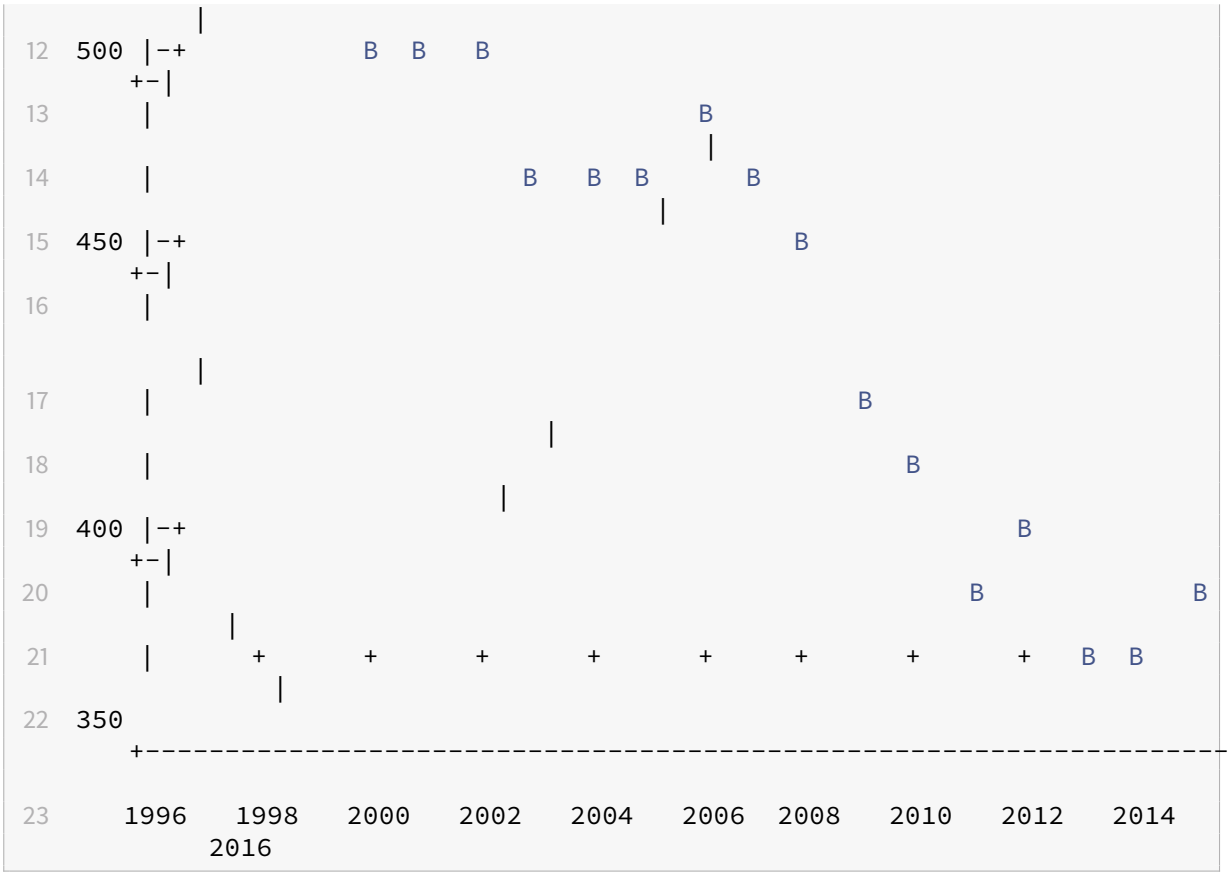
```
cat crime_rate | awk '{print $1-$2}'
```

Summary stats, using `st`

```
cat crime_rate | awk '{print $1}' | st
# N       min     max     sum     mean    stddev
# 20      1997    2016    40130   2006.5  5.91608
```

Let's plot it

```
$ gnuplot -p -e "set terminal dumb; plot 'crime_rate' with points pt 2"
650
    +------------------------------------------------------------------------+
     |      +      +      +      +      +     +      +      +      +
       |
     |                                              'crime_rate'    B
          |
600 |-+B
    +-|
     |
         |
     |      B                                                          |
     |
         |
550 |-+
    +-|
     |          B                                                          |
     |
```


500
450
400
350
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016