The Opinionated Guide to sed

2017-05-05

Most tools in the UNIX kit have a line-based orientation, i.e. ASCII LF is used as the record-separator — not ASCII RS. As a stream editor, sed can be used in many different ways, however its major use is in re-shaping data to conform to UNIX's line-orientation.

Some people will attempt to shoehorn sed into other solutions, usually when working with complex structured data. “Derp derp... just gonna split this CSV on the commas, that will always work.”

Never use regular expressions to work with escape-able separators. Failure & Madness will perch upon your shoulders.

The two basic models for reshaping data with sed are either a) expand a line, or b) condense a paragraph.

Expanding GrandStream Phone Configs

These configs look like *P23=foo&P24=bar*. To identify the variable with the value we're searching for:

```
sed 's/&/\n/g' < config_file | grep -a foo
```

Pretty easy, right?

Condensing

Condensing a paragraph into a line is not much more complex, we just need to use sed's hold space.

The basic pattern for condensing is:

```
push every line into the hold space
when we match the end of the paragraph, then
  exchange the pattern and hold spaces,
  replace all the newlines with spaces,
  and print.
```

Condensing Postfix Queues

Postfix queue listings look like:

```
-Queue ID- --Size-- ----Arrival Time---- -Sender/Recipient-------
D36F3304D5A9     8432 Sun May  8 11:51:13  MAILER-DAEMON
              (connect to kbslave.com[211.23.19.130]:25:  Connection timed out)
                                           noreply@kbslave.com
```

where the bounce-back email address and queue IDs are on separate lines.

To print each queue entry one per line:

```
postqueue -p | sed -n '
1d
H
/^$/ {
    x
    s/\n//g
    p
}'
```

Now we can easily dump all the queued mail for a certain address by piping the reshaped postfix queue output into:

```
| awk -v "bad_addr=bad@bad.org" '
$NF == bad_addr { system(sprintf("postsuper -d %s", $1)) }
'
```

Condensing Wireshark Packet Captures

tshark will dump the whole packet, while we only want the high-level HTTP data. One solution is to crank up the scrollback of your tmux, another is to just output the lines that we care about.

```
tshark -n -i eth0 -ql -V port 80 | sed -n '
s/\r//g
H
/^$/ {
    x
    /Hypertext/ {
        s/.*\(Hypertext\)/\n\1/
        p
    }
}'
```

Note the -l argument to tshark; the default action is to buffer all output, which is not good for this use case. -l will make the output line-buffered.

The -q is a nicety, we don't want output cluttered with stderr counts of how many packets we've seen.

Auditing iSCSI with Wireshark

We can also use sed to quickly diagnose connection issues.

```
tshark -nl i eth0 port 3260 and host ${initiator} | sed '
/iSCSI/!d
/\[TCP ACKed unseen segment/d
/\[/p
s/.*\(iSCSI.*\)/\1/
/SCSI: .* (Good)/d
/SCSI: Data In LUN:/d
/SCSI: Data Out LUN:/d
/SCSI Data Out$/d
/SCSI: Read[^ ]* LUN:/d
/SCSI: Synchronize Cache/d
/iSCSI [^ ]* Ready To Transfer/d
'
```

tshark will note any TCP issues in square brackets, so just drop any other high-volume normal traffic. NOPs are not deleted as we want to see the keepalives.

Applying this sed filter makes it quite clear when iSCSI issues arise.

```
iSCSI 114 NOP In
iSCSI 114 NOP Out
iSCSI 114 NOP In
9522 37.512569215   10.81.0.24 -> 10.81.0.16   iSCSI 114 [TCP Retransmission]
    SCSI: Synchronize Cache(10) LUN: 0x2c (LBA: 0x00000000, Len:  0)
```

ISC DHCPd

Say we want to find the config stanza for a given host's DHCP lease. Each lease in ISC dhcpd's lease file looks something like:

```
lease 10.75.127.212 {
  starts 5 2016/03/18 00:31:38;
  ends 5 2016/03/18 00:33:28;
  tstp 5 2016/03/18 00:33:28;
  cltt 5 2016/03/18 00:31:38;
  binding state free;
  hardware ethernet 6c:8d:c1:8a:2e:ef;
}
```

grep'ing for the MAC address will never get us the IP address, so a common solution is to grep -B 10, which usually gets us the lease. However, since each lease can have a variable number of line items, it's not ideal.

The following sed script turns each stanza into one line:

```
#n
/^#/ b
H
/^}/ {
    s/.*//
    x
    s/\n/ /g
    p
}
```

where `#n` in a sed script file means the same as sed -n, and `/^#/ b` stops processing comment lines by branching to the empty/null label.

Conclusion

The above sed scripts have all been shown in expanded form. Each line-terminating ASCII LF can be replaced by a semi-colon. E.g.

```
postqueue -p | sed -n '1d; H; /^$/ { x; s/\n//g; p }'
```