Lecture #16 – Advanced SED (Chapter 5)

- Hold, Get, and Exchange

  Sed has 2 buffers, the "pattern" buffer and the "holding" buffer.

  When sed reads a line from the input file, it is placed into the "pattern" buffer. Commands are executed against the "pattern" buffer.

  Lines can be sent from the "pattern" to the "holding" buffer using "h" or "H" These lines can be later retrieved with the "g" or "G" command.

  h copy line from pattern buffer to holding buffer
  H append line from pattern buffer to holding buffer

  g copy line from holding buffer to pattern buffer replacing the current line
  G copy line from holding buffer to pattern buffer appending to the current line

  x exchange the line in the holding buffer with the pattern buffer

  Examples:

  Append lines containing "northeast" to the end of the file
  $ sed –e '/northeast/h' -e '$G' datafile

  Take the line containing "northeast" and use it to replace the last line.
  $ sed –e 'northeast/h' –e '$g' datafile

  Place lines containing "NW" after line containing "WE"
  $ sed –e '/NE/{h; d; }' –e '/WE/{G; }' datafile

  Replace the line containing Margot with the line containing Patricia
  $ sed –e '/Patricia/h' –e '/Margot/x' datafile

- Nesting commands

  Commands can be nested in SED using curly braces.

  Example:

```
/southwest/, /southeast/ {      # in blocks between western and southeast
        /^ *$/d                 # delete blank lines
        /Susan/ { h; d; }       # copy and remove lines with "Susan"
}
/Ann/g                          # replace lines with "Ann" from holding buffer
s/TB \(Savage\)/Thomas \1/      # replace TB Savage with Thomas Savage
```

- Multi-line commands

  D delete through first embedded newline in pattern buffer
  P print through first embedded newline in pattern buffer
  N similar to "n", read next input line, but append to current pattern buffer

- Branching

  You can label lines in a sed program by using a colon followed by a label name. The 'b' command can be used to branch to a given label.

  Example (join together lines ending with backslashes):

```
:b
/\\$/ {                 # lines ending with backslash
        N               # append next line to pattern space
        s/\\\n//        # remove backslash and newline
        bb              # jump to label "b"
}
```

- Example (simulating uniq):

```
:b
$b                      # on last line, branch to end

N                       # append next line to pattern space
/^\(.*\)\n\1$/ {        # does next line match previous one?
        s/.*\n//        # remove the previous line
        bb              # jump to label "b"
}

$b                      # on last line, branch to end
P                       # print and delete first line in pattern space
D
```

- Example (delete comments from C code):

```
/\/\*/!n                # skip lines with comments

:x
/\*\//! {               # append until we get the trailing side of comment
        N
        bx
}

s/\/\*.*\*\///          # remove between the comment delimiters
```

- SED One-liners

  Double space input file
  $ sed 'G' datafile

  Double space input file that already has blank lines
  $ sed '/^$/d; G' datafile

  Insert a blank line above and below every line that matches "regexp"
  $ sed '/regexp/ {x; p; x; G}' datafile

  Print each line with line numbers
  $ sed = datafile | sed 'N; s/\n/\t/'

  Print number of lines in input file (i.e. wc –l)
  $ sed –n '$=' datafile

  Delete leading and trailing whitespace from all lines
  $ sed 's/^[\t]*//; s/[\t]*$//' datafile

  Print 1st line of a file (i.e. head -1)
  $ sed 'q' datafile

  Print last line of a file (i.e. tail -1)
  $ sed –n '$p' datafile

- Reverse the lines in a file

```
#!/usr/bin/sed -nf

# reverse all lines of input, i.e. first line became last, ...

# first line is pasted into buffer
1{h;b;}

# for all other lines, the buffer (which contains all previous)
# is appended to current line, so, the order is being reversed
# on the buffer, after that is done, store everything on the buffer
# again
G;h

# the last line (after have done the above job) get the contents
# of buffer, and print it
${g;p;}
```

- Reversing characters in every line

```
# Ignore empty lines
/./!n

# Escape ! by doubling it, place -!- markers at begin and end of line
s/!/!!/g
s/^/-!-/
s/$/-!-/

# Swap first char after beginning marker with first char before ending marker, repeat
ta
:a
s/-!-\([^!]\|!!\)\(.*\)\)[^!]\|!!\)-!-/\3-!-\2-!-\1/
ta

# Delete markers, and unescape exclamation points
s/-!-//g
s/!!/!/g
```