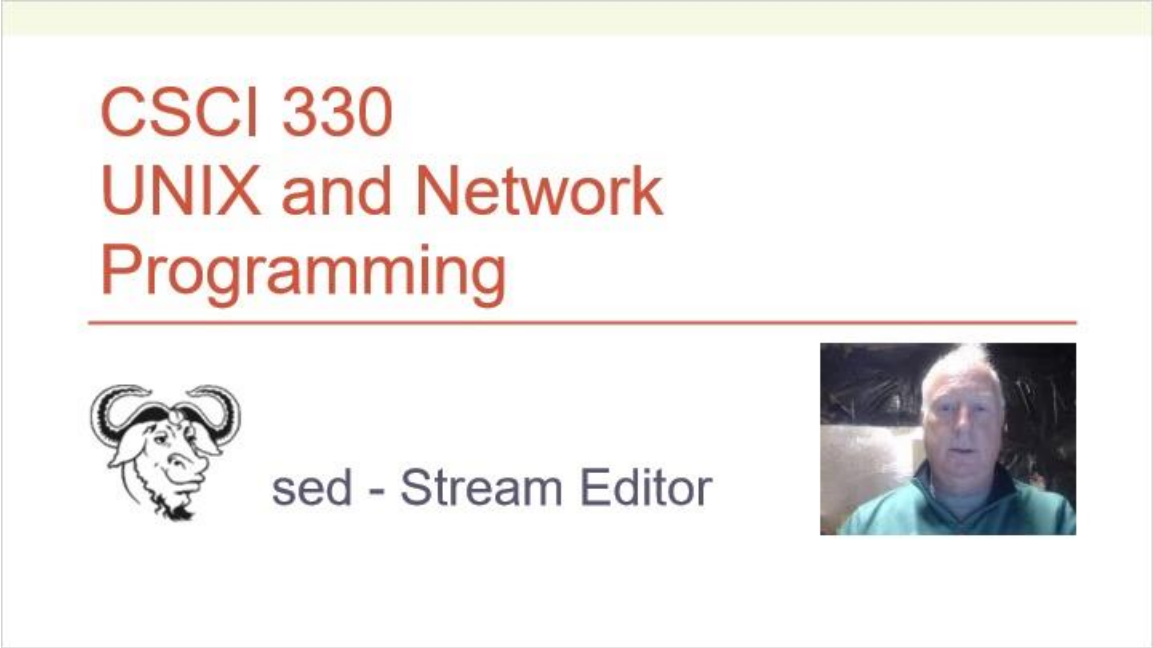
Stream Editor

1. Stream Editor

1.1 Introduction



1.2 What is sed?

What is sed?

- A non-interactive stream editor
- Use sed to:
  - Automatically perform edits on file(s)
  - Simplify doing the same edits on multiple files
  - Write conversion programs
  - Do editing operations from shell script

1.3 sed command syntax

sed command syntax

```
$ sed -e 'address command' input_file
```

(a) Inline Script

```
$ sed -f script.sed input_file
```

(b) Script File

1.4 How Does sed Work?

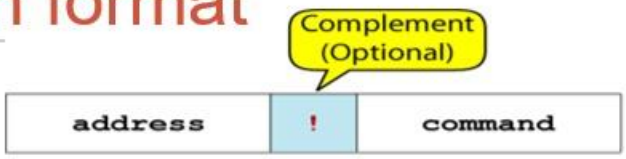
How Does sed Work?

- sed reads file line by line
  - line of input is copied into a temporary buffer called pattern space
  - editing instructions are applied to line in the pattern space
  - line is sent to output (unless "-n" option was used)
  - line is removed from pattern space
- sed reads next line of input, until end of file

Note: input file is unchanged unless "-i" option is used

1.5 sed instruction format

sed instruction format



- address determines which lines in the input file are to be processed by the command(s)
  - if no address is given, then the command is applied to each input line
- address types:
  - Single-Line address
  - Set-of-Lines address
  - Range address

1.6 Single-Line Address

Single-Line Address

- Specifies only one line in the input file
  - special: dollar sign ($) denotes last line of input file

Examples:

- show only line 3

```
sed -n -e "3 p" inFile
```

- show only last line

```
sed -n -e '$ p' inFile
```

- substitute “endif” with “fi” on line 10

```
sed -e "10 s/endif/fi/" inFile
```

1.7 Set-of-Lines Address

Set-of-Lines Address

- use regular expression to match lines
  - written between two slashes
  - process only lines that match
  - may match several lines
  - lines don’t have to be consecutive

Examples:

```
sed -i -e "/key/ s/more/other/" inFile
sed -n -e "/r..t/ p" input-file
```

1.8 Range Address

Range Address

- Defines a set of consecutive lines

Format:

startAddr,endAddr (inclusive)

Examples:

10,50	line-number,line-number
10,/funny/	line-number,/RegExp/
/funny/,10	/RegExp/,line-number
/funny/,/sad/	/RegExp/,/RegExp/

1.9 Example: Range Address

Example: Range Address

```
% sed -n -e '/Line 2/,/Line 4/p' inFile
```

start end

- print lines 2 to 4, inclusive:

```
Line 1 of input
Line 2 of input
Line 3 of input
Line 4 of input
Line 5 of input
```

These lines are printed

1.10 Address with !

Address with !

- address with an exclamation point (!):
  command applies to lines that do not match the address

Example:
print lines that do not contain “obsolete”

```
sed -n -e '/obsolete/!p' inFile
```

1.11 sed Commands

sed Commands

- modify
  - insert, append, change
  - delete
  - substitute
- I/O
  - next, print
  - read, write
- quit

1.12 Command: i, a, c

Command: i, a, c

- "i" adds line(s) before the address
- "a" adds line(s) after the address
- "c" replaces an entire matched line with new text

Syntax:

```
[address] i\
text
```

1.13 Example: Insert Command (i)

Example: Insert Command (i)

```
% cat tut.insert.sed
1 i\
        Tuition List\

% cat tuition.data
Part-time        1003.99
Two-thirds-time  1506.49
Full-time        2012.29
% sed -f tut.insert.sed tuition.data
        Tuition List

Part-time        1003.99
Two-thirds-time  1506.49
Full-time        2012.29
```

sed script to insert "Tuition List" as report title before line 1

input data

output after applying the insert command

1.14 Delete Command: d

Delete Command: d

- deletes the entire pattern space
  - commands following the delete command are ignored since the deleted text is no longer in the pattern space

Syntax:

```
[address] d
```

1.15 Substitute Command (s)

Substitute Command (s)

Syntax:

```
[address] s/search/replacement/[flag]
```

- replaces text “search” string with “replacement” string
- “search” & “replacement string” can be regular expression
- flag:
  - specific substitution count (integer), default “1”
  - global (“g”), i.e. replace all occurrences

1.16 Example: substitute word in file

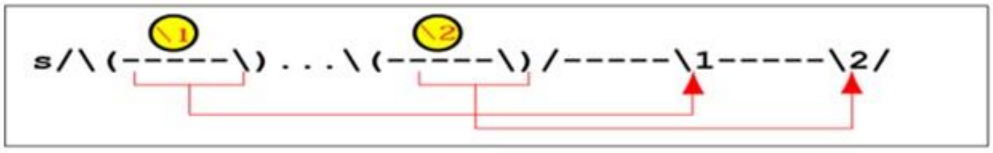
Example: substitute word in file

```
$ cat myFile
This is a lively liitle text file
With quite a few words that make
Up a good assembly of lines

$ sed -i -e "s/liitle/little/g" myFile

$ cat myFile
This is a lively little text file
With quite a few words that make
Up a good assembly of lines
```

1.17 Substitution Back References

Substitution Back References

(a) Whole Pattern Substitution



(b) Numbered Buffer Substitution

1.18 Example: Replacement String &

Example: Replacement String &

```
$ cat datafile
Charles Main            34
Patricia Jones          7
TB Savage               20
Margot Weber            9
Ann Stephens            13

$ sed -e 's/[0-9][0-9]$/&.5/g' datafile
Charles Main            34.5
Patricia Jones          7
TB Savage               20.5
Margot Weber            9
Ann Stephens            13.5
```

1.19 Example: Back Reference

Example: Back Reference

```
$ cat name.data
John Doe
Susan Maloney
Harvey Keitel
Randy Newman
Ossie Weaver

$ sed -e 's/\(\<.*\>\) \(\<.*\>\)/\2, \1/g' name.data
Doe, John
Maloney, Susan
Keitel, Harvey
Newman, Randy
Weaver, Ossie
```

1.20 I/O Commands: n and p

I/O Commands: n and p

- n (lowercase)
  - copies the contents of the pattern space to output
  - deletes the current line in the pattern space
  - refills it with the next input line
  - continue processing
- p (lowercase)
  - copies the entire contents of the pattern space to output
  - will print same line twice unless the option “-n” is used

1.21 File commands

File commands

- allows to read and write from/to file while processing standard input
- r read command
- w write command

1.22 quit (q) Command

quit (q) Command

Syntax: `[addr]q`

- Quit (exit sed) when addr is encountered

Example: Display the first 50 lines and quit

```
% sed -e "50q" datafile
```

1.23 Summary: stream editor

Summary: stream editor

- can be called from shell script
- allows systematic wholesale changes to files