# CSCI 330
# THE UNIX SYSTEM

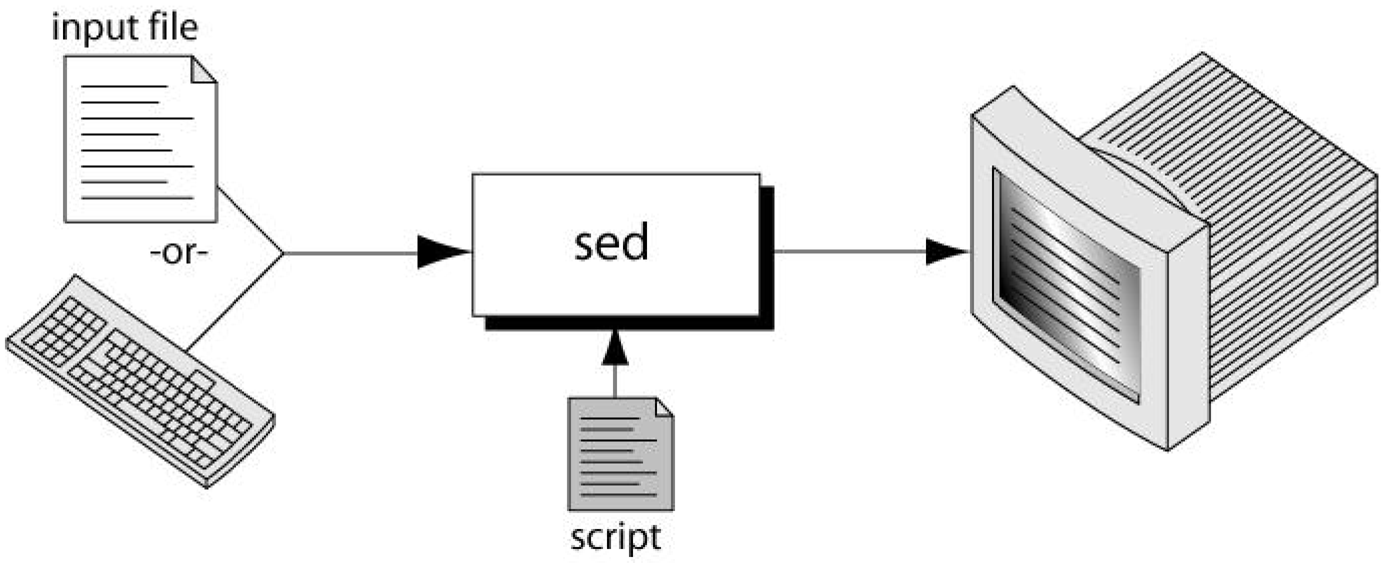
sed - Stream Editor

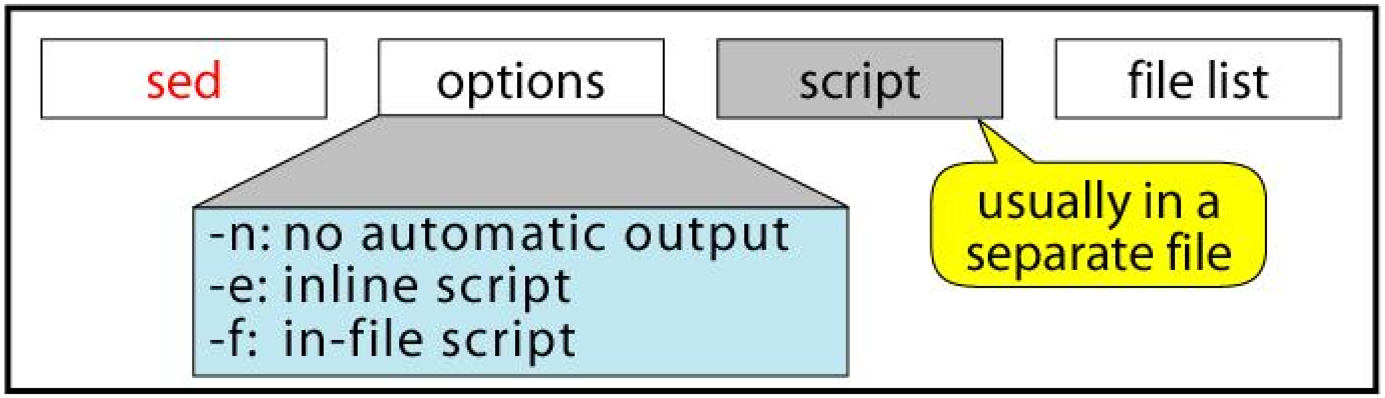
## WHAT IS SED?

- A non-interactive stream editor
- Interprets sed instructions and performs actions

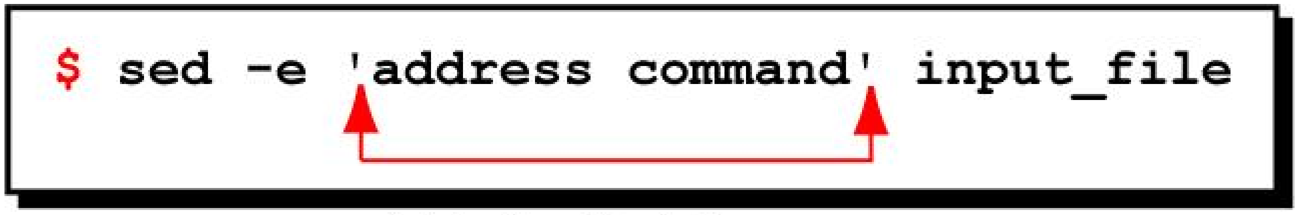
- Use sed to:
  - Automatically perform edits on file(s)
  - Simplify doing the same edits on multiple files
  - Write conversion programs

## THE SED COMMAND



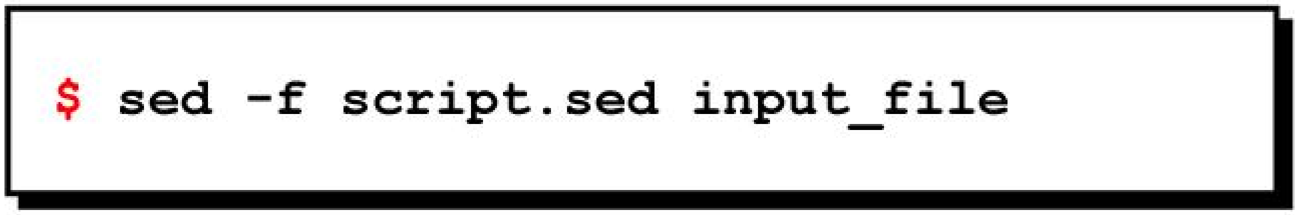


## SED COMMAND SYNTAX



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

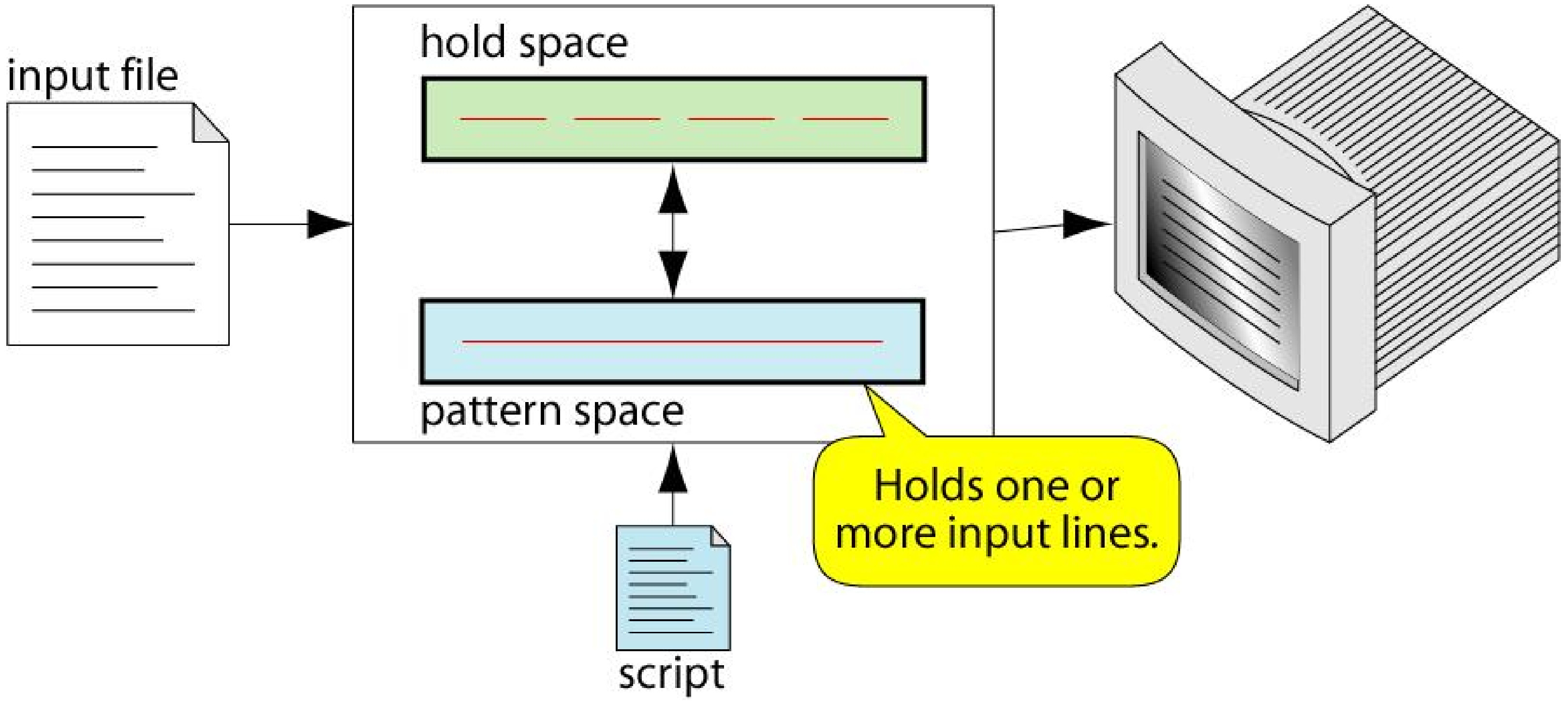
(a) Inline Script



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

(b) Script File

## SED OPERATION



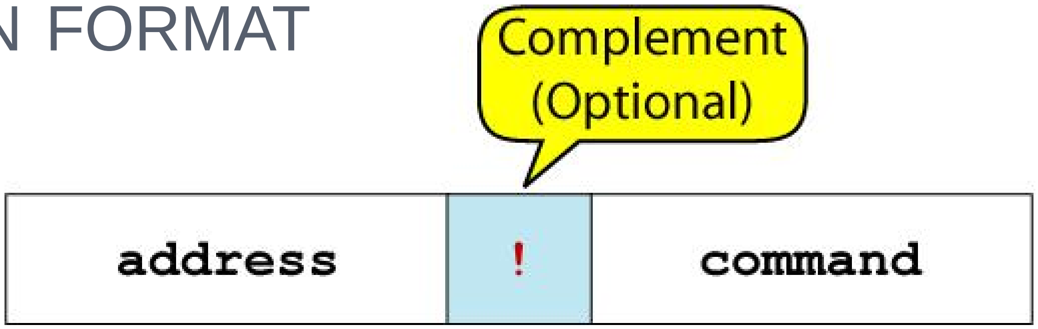
## How Does sed Work?

- sed reads line of input
    - line of input is copied into a temporary buffer called pattern space
    - editing commands are applied
        - subsequent commands are applied to line in the pattern space, not the original input line
        - once finished, 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

## SED INSTRUCTION FORMAT



- address determines which lines in the input file are to be processed by the command(s)
  - if no address is specified, then the command is applied to each input line
- address types:
  - Single-Line address
  - Set-of-Lines address
  - Range address
  - Nested 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' input-file
```
- show only last line
```
sed -n -e '$ p' input-file
```
- substitute "endif" with "fi" on line 10
```
sed -e '10 s/endif/fi/' input-file
```

## SET-OF-LINES ADDRESS

- use regular expression to match lines
  - written between two slashes
  - process only lines that match
  - may match several lines
  - lines may or may not be consecutives

Examples:

```
sed -e ‘/key/ s/more/other/’ input-file
sed -n -e ‘/r..t/ p’ input-file
```

## Range Address

- Defines a set of consecutive lines

Format:

start-addr,end-addr (inclusive)

Examples:

| | |
|---|---|
| 10,50 | line-number,line-number |
| 10,/R.E/ | line-number,/RegExp/ |
| /R.E./,10 | /RegExp/,line-number |
| /R.E./,/R.E/ | /RegExp/,/RegExp/ |

## Example: Range Address

```
% sed -n -e '/^BEGIN$/,/^END$/p' input-file
```

addr1

addr2

- Print lines between BEGIN and END, inclusive

```
BEGIN
Line 1 of input
Line 2 of input
Line3 of input
END
Line 4 of input
Line 5 of input
```

These lines are printed

## Nested Address

- Nested address contained within another address

Example:

print blank lines between line 20 and 30

```
20,30{
  /^$/ p
}
```

## Address with !

- address with an exclamation point (!):

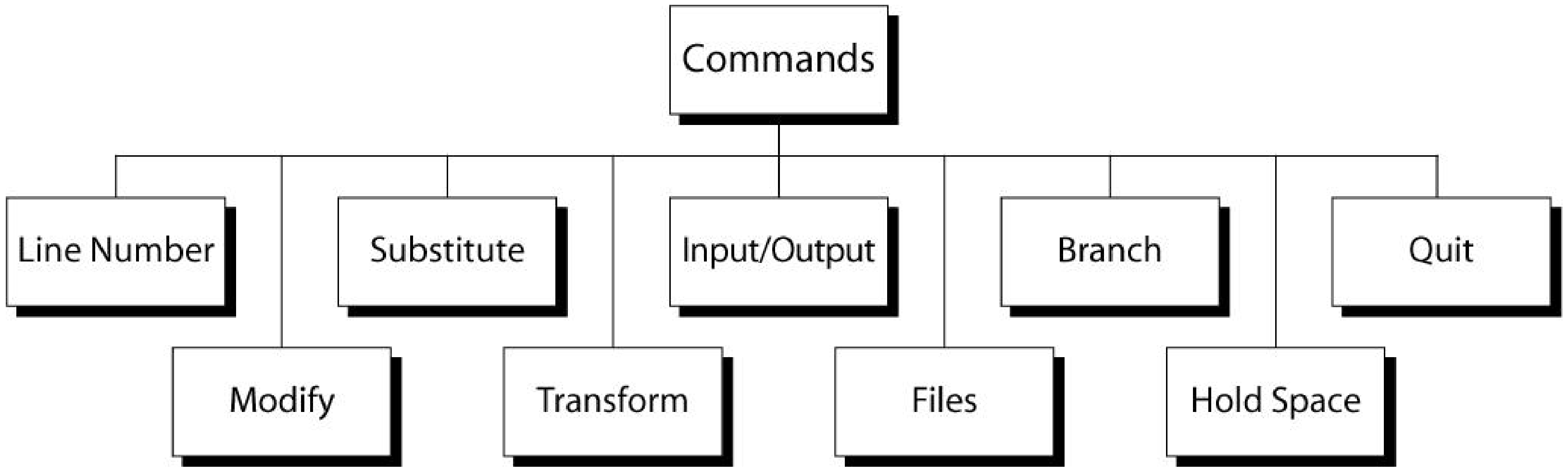
  instruction will be applied to all lines that do not match the address

<u>Example:</u>

print lines that do not contain “obsolete”

```
sed -e ‘/obsolete/!p’ input-file
```

## SED COMMANDS



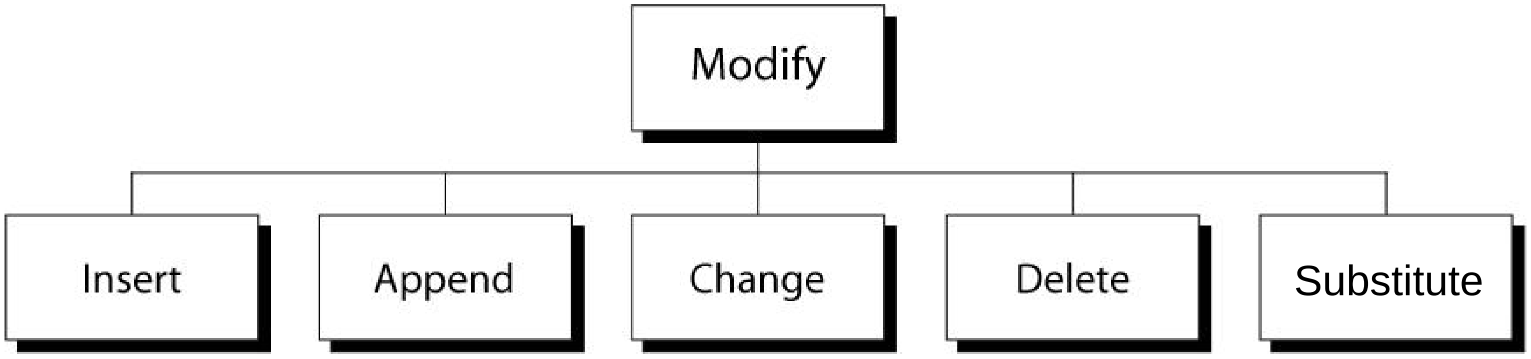
## Line Number

- line number command (=) writes the current line number before each matched/output line

Examples:

```
sed -e '/Two-thirds-time/=' tuition.data
sed -e '/^[0-9][0-9]/=' inventory
```

## MODIFY COMMANDS



## INSERT COMMAND: I

- adds one or more lines directly to the output before the address:
  - inserted “text” never appears in sed’s pattern space
  - cannot be used with a range address; can only be used with the single-line and set-of-lines address types

<u>Syntax:</u>

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

## EXAMPLE: INSERT COMMAND (I)

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

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

## APPEND COMMAND: A

- adds one or more lines directly to the output after the address:
  - Similar to the insert command (i), append cannot be used with a range address.
  - Appended “text” does not appear in sed’s pattern space.

Syntax:

```
[address] a\
text
```

## EXAMPLE: APPEND COMMAND (A)

```
% cat tuition.append.sed
a \
--------------------------
% cat tuition.data
Part-time          1003.99
Two-thirds-time    1506.49
Full-time          2012.29
% sed -f tuition.append.sed tuition.data
Part-time          1003.99
--------------------------
Two-thirds-time    1506.49
--------------------------
Full-time          2012.29
--------------------------
```

Sed script to append dashed line after each input line

Input data

Output after applying the append command

## Change Command: c

- replaces an entire matched line with new text
- accepts four address types:
  - single-line, set-of-line, range, and nested addresses.

<u>Syntax:</u>

```
[address1[,address2]] c\
text
```

## EXAMPLE: CHANGE COMMAND (C)

```
% cat tuition.change.sed
1 c\
Part-time         1100.00
% cat tuition.data
Part-time         1003.99
Two-thirds-time   1506.49
Full-time         2012.29
% sed -f tuition.change.sed tuition.data
Part-time         1100.00
Two-thirds-time   1506.49
Full-time         2012.29
```

Sed script to change tuition cost from 1003.99 to 1100.00

Input data

Output after applying the change command

## 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:

```
[address1[,address2]] d
```

## Example: Delete Command (d)

- Remove part-time data from “tuition.data” file

```
% cat tuition.data
Part-time         1003.99
Two-thirds-time   1506.49
Full-time         2012.29
```

$\}$ Input data

```
% sed –e '/^Part-time/d' tuition.data
Two-thirds-time   1506.49
Full-time         2012.29
```

$\}$ Output after applying delete command

## SUBSTITUTE COMMAND (S)

Syntax:

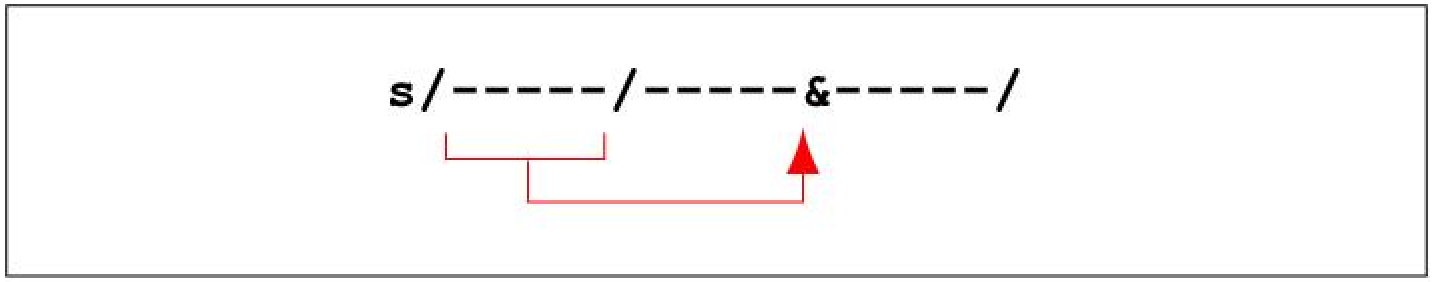
```
[addr1][,addr2] s/search/replace/[flags]
```

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

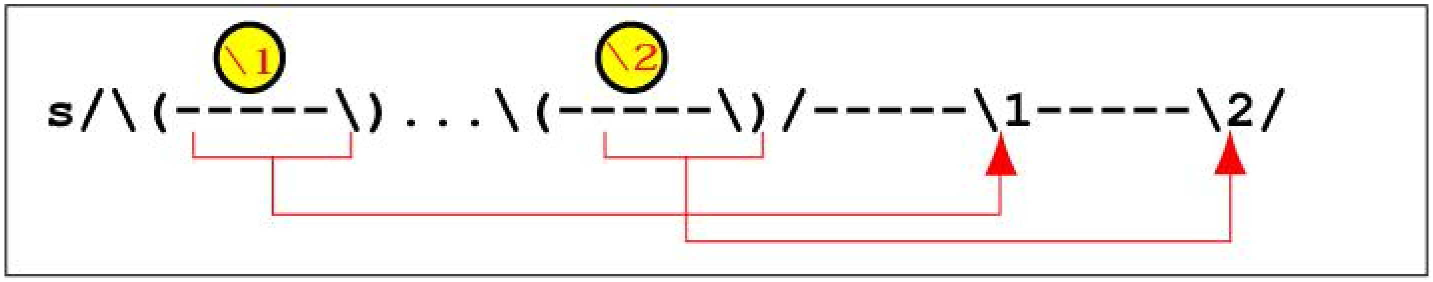
## Regular Expressions: use with sed

| Metacharacter | Description/Matches… |
|---|---|
| **.** | Any one character, except new line |
| ***** | Zero or more of preceding character |
| ^ | A character at beginning of line |
| **$** | A character at end of line |
| **\char** | Escape the meaning of *char* following it |
| **[ ]** | Any one of the enclosed characters |
| \( \) | Tags matched characters to be used later |
| **x\{m\}** | Repetition of character x, m times |
| \< | Beginning of word |
| \> | End of word |

## SUBSTITUTION BACK REFERENCES



(a) Whole Pattern Substitution



(b) Numbered Buffer Substitution

## EXAMPLE: REPLACEMENT STRING &

```
$ cat datafile
Charles Main        3.0    .98    3       34
Sharon Gray         5.3    .97    5       23
Patricia Hemenway   4.0    .7     4       17
TB Savage           4.4    .84    5       20
AM Main Jr.         5.1    .94    3       13
Margot Weber        4.5    .89    5        9
Ann Stephens        5.7    .94    5       13

$ sed -e ‘s/[0-9][0-9]$/&.5/’ datafile
Charles Main        3.0    .98    3       34.5
Sharon Gray         5.3    .97    5       23.5
Patricia Hemenway   4.0    .7     4       17.5
TB Savage           4.4    .84    5       20.5
AM Main Jr.         5.1    .94    3       13.5
Margot Weber        4.5    .89    5        9
Ann Stephens        5.7    .94    5       13.5
```

## Example: Back Reference

```
$ cat filedata
/home/ux/user/z156256
/home/ux/user/z056254
/home/lx/user/z106253
/home/ux/user/z150252
/home/mp/user/z056254
/home/lx/user/z106253

$ sed -e 's,/home/\(..\)/user/\(z[0-9]\{6\}\),/usr/\2/\1,g' filedata
/usr/z156256/ux
/usr/z056254/ux
/usr/z106253/lx
/usr/z150252/ux
/usr/z056254/mp
/usr/z106253/lx
```

## Transform Command (y)

Syntax:

```
[addr1][,addr2]y/a/b/
```

- translates one character 'a' to another 'b'
- cannot use regular expression metacharacters
- cannot indicate a range of characters
- similar to “tr” command

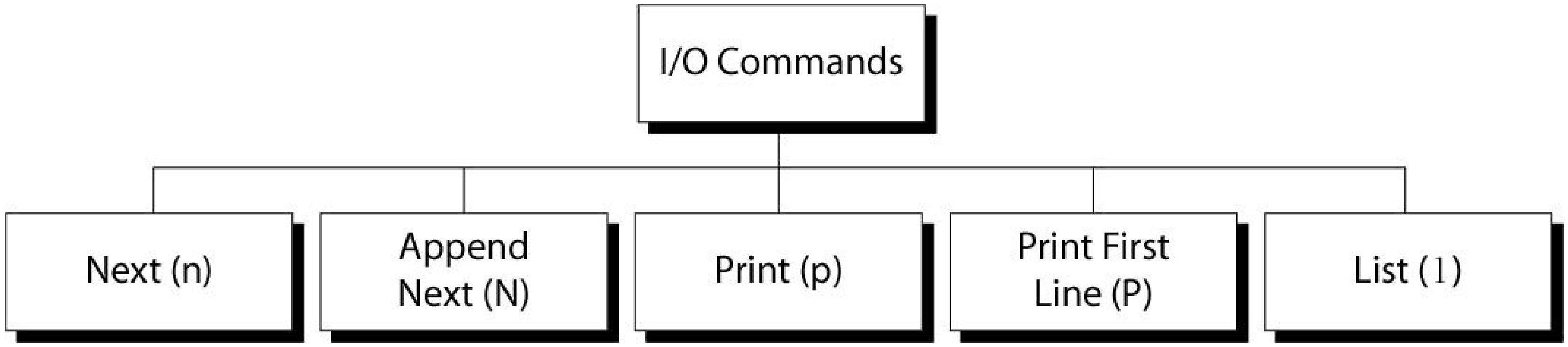
Example:

```
$ sed -e ‘1,10y/abcd/wxyz/’ datafile
```

$\underbrace{\underbrace{\texttt{abcd}}\texttt{/}\underbrace{\texttt{wxyz}}}$

Must have same number of characters

## SED I/O COMMANDS



## Input (next) Command: n and N

- Forces sed to read the next input line
  - 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
- N (uppercase) Command
  - adds the next input line to the current contents of the pattern space
  - useful when applying patterns to two or more lines at the same time

## Output Command: p and P

- Print Command (p)
  - copies the entire contents of the pattern space to output
  - will print same line twice unless the option "–n" is used
- Print command: P
  - prints only the first line of the pattern space
  - prints the contents of the pattern space up to and including a new line character
  - any text following the first new line is not printed

## List Command (l)

- The list command: l
  - shows special characters (e.g. tab, etc)

- The octal dump command (od -c) can be used to produce similar result

## Hold Space

- temporary storage area

  used to save the contents of the pattern space

- 4 commands that can be used to move text back and forth between the pattern space and the hold space:

```
h, H
g, G
```

## Hold Commands: h and H

- The lowercase hold (and replace) command (h) copies the current contents of the pattern space to the hold space and replaces any text currently in the hold space
- The uppercase hold (and append) command (H) appends the current contents of the pattern space to the hold space

## The Get Commands: g and G

- The lowercase get (and replace) command (g) copies the text in the hold space to the pattern space and replaces any text currently in the pattern space
- The uppercase get (and append) command (G) appends the current contents of the hold space to the pattern space

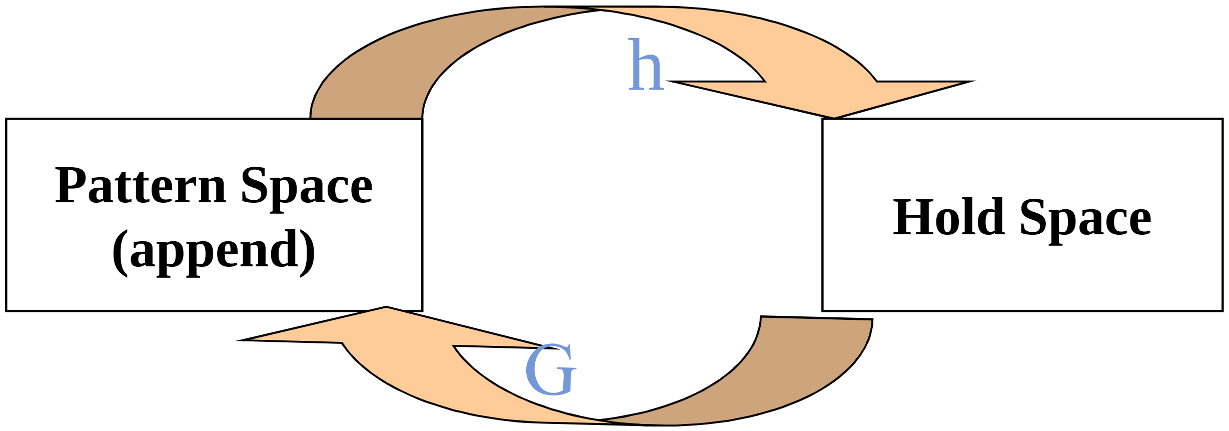
## The 'h' and 'G' Commands

Syntax: **`[addr1][,addr2]h`**

- copies the contents of the pattern space to a hold space; replaces any text currently in the hold space

Syntax: **`[addr1][,addr2]G`**

- gets what was in the hold space and copies it into the pattern space, appending to what was there



## Example: The ‘h’ and ‘G’ Commands

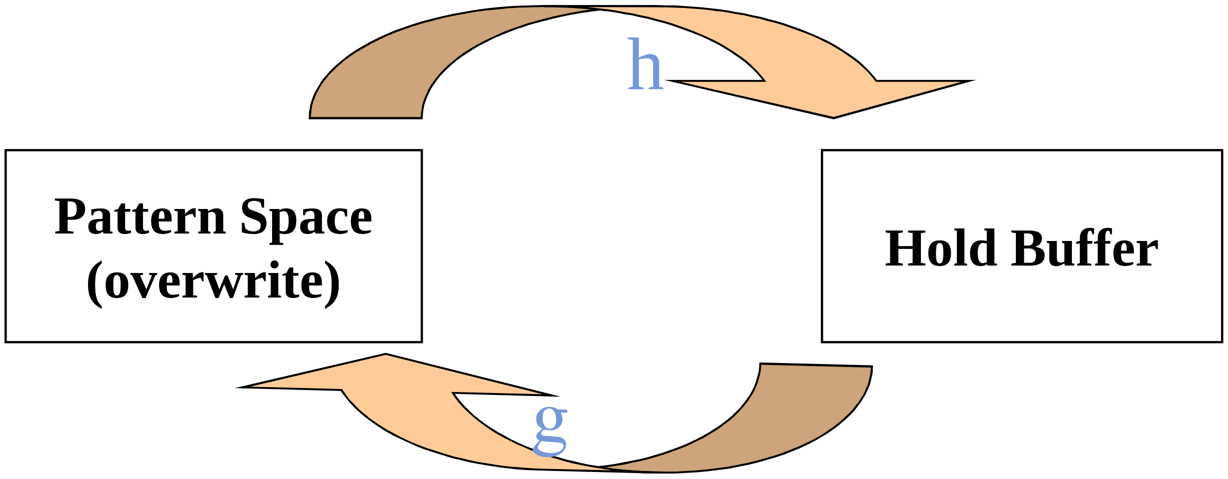
```
% sed -e ‘/northeast/h’ -e ‘$G’ datafile
northwest        NW
western          WE
southwest        SW
southern         SO
southeast        SE
eastern          EA
northeast        NE
north            NO
central          CT
northeast        NE
```

```
% cat datafile
northwest        NW
western          WE
southwest        SW
southern         SO
southeast        SE
eastern          EA
northeast        NE
north            NO
central          CT
```

## The ‘g’ Command

Syntax: **`[addr1][,addr2]g`**

- Gets what was in the hold space and copies it into the pattern space, overwriting what was there



## Example: The ‘h’ and ‘g’ Commands

```
% sed -e ‘/northeast/h’ -e ‘$g’ datafile
northwest        NW
western          WE
southwest        SW
southern         SO
southeast        SE
eastern          EA
northeast        NE
north            NO
northeast        NE
```

```
% cat datafile
northwest        NW
western          WE
southwest        SW
southern         SO
southeast        SE
eastern          EA
northeast        NE
north            NO
central          CT
```

## File commands

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

## READ FILE COMMAND

Syntax: **`r filename`**

- queue the contents of filename to be read and inserted into the output stream at the end of the current cycle, or when the next input line is read
  - if filename cannot be read, it is treated as if it were an empty file, without any error indication

- single address only

## WRITE FILE COMMAND

Syntax: **`w filename`**

- Write the pattern space to filename
- The filename will be created (or truncated) before the first input line is read
- all w commands which refer to the same filename are output through the same FILE stream

## Branch Command (b)

- Change the regular flow of the commands in the script file

Syntax: **`[addr1][,addr2]b[label]`**

  - Branch (unconditionally) to 'label' or end of script
  - If "label" is supplied, execution resumes at the line following :label; otherwise, control passes to the end of the script

- Branch label

```
:mylabel
```

•Can be up to 7 characters
•Must be on a line by itself
•Must begin with a colon
•No spaces after it and after the colon

## EXAMPLE: BRANCH (B) COMMAND

<u>Example:</u>

- If the string 'soph' is found on a line, write the matched line to a file called "soph.students"; otherwise, write unmatched lines to a file called 'others':

```
/soph/b save
w others
b
:save
w soph.students
```

## EXAMPLE: THE 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
```

Same as:

```
% sed -n -e ‘1,50p’ datafile
% head -50 datafile
```