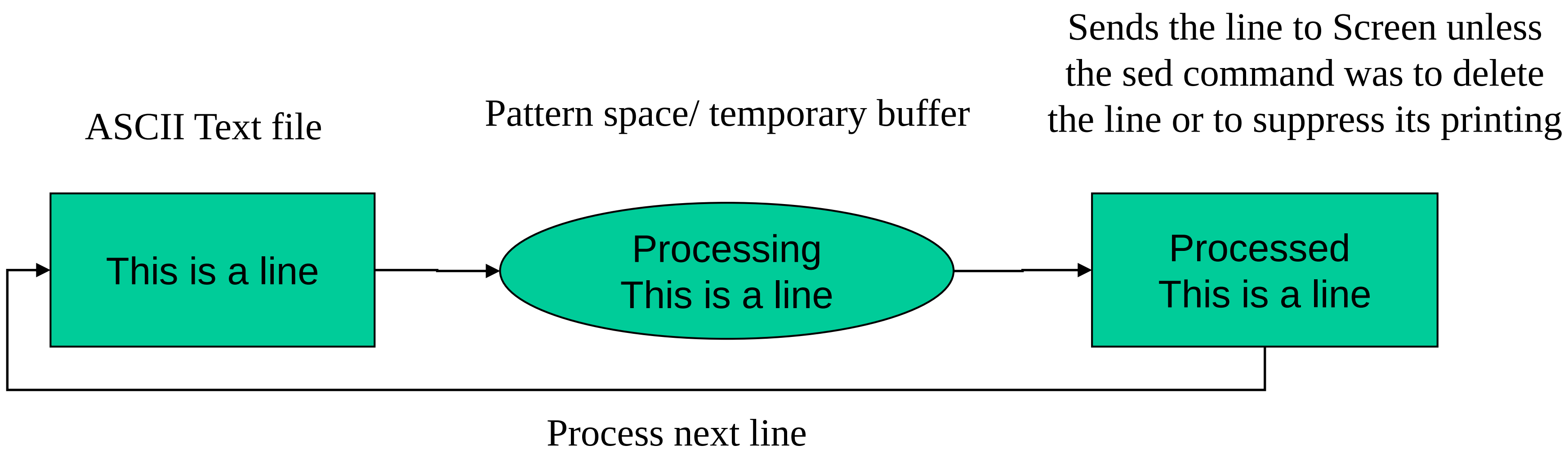
SED —Streamlined EDitor

- A streamlined, non-interactive editor
- All lines are printed to the screen by default
- Non destructive: the original file is never altered or destroyed by default (there is an option -i to change this)
- GNU version of `sed` and standard UNIX distribution: `sed --version`

How Does SED Work?


Sends the line to Screen unless the sed command was to delete the line or to suppress its printing
ASCII Text file
Pattern space/ temporary buffer
This is a line
Processing
This is a line
Processed
This is a line
Process next line

- Search for pattern matched lines in the text file using
  - Regular expressions with meta characters (special characters)
- Processing text file line by line according to the `sed` commands
- Sends the processed line to standout

sed Usage

- Command line options
  - `sed [-options] [-f file] [input-files]`
    - `-f`: tells sed to get commands from a file (sed script)
  - `sed [-optins] ‘commands’ <input files>`

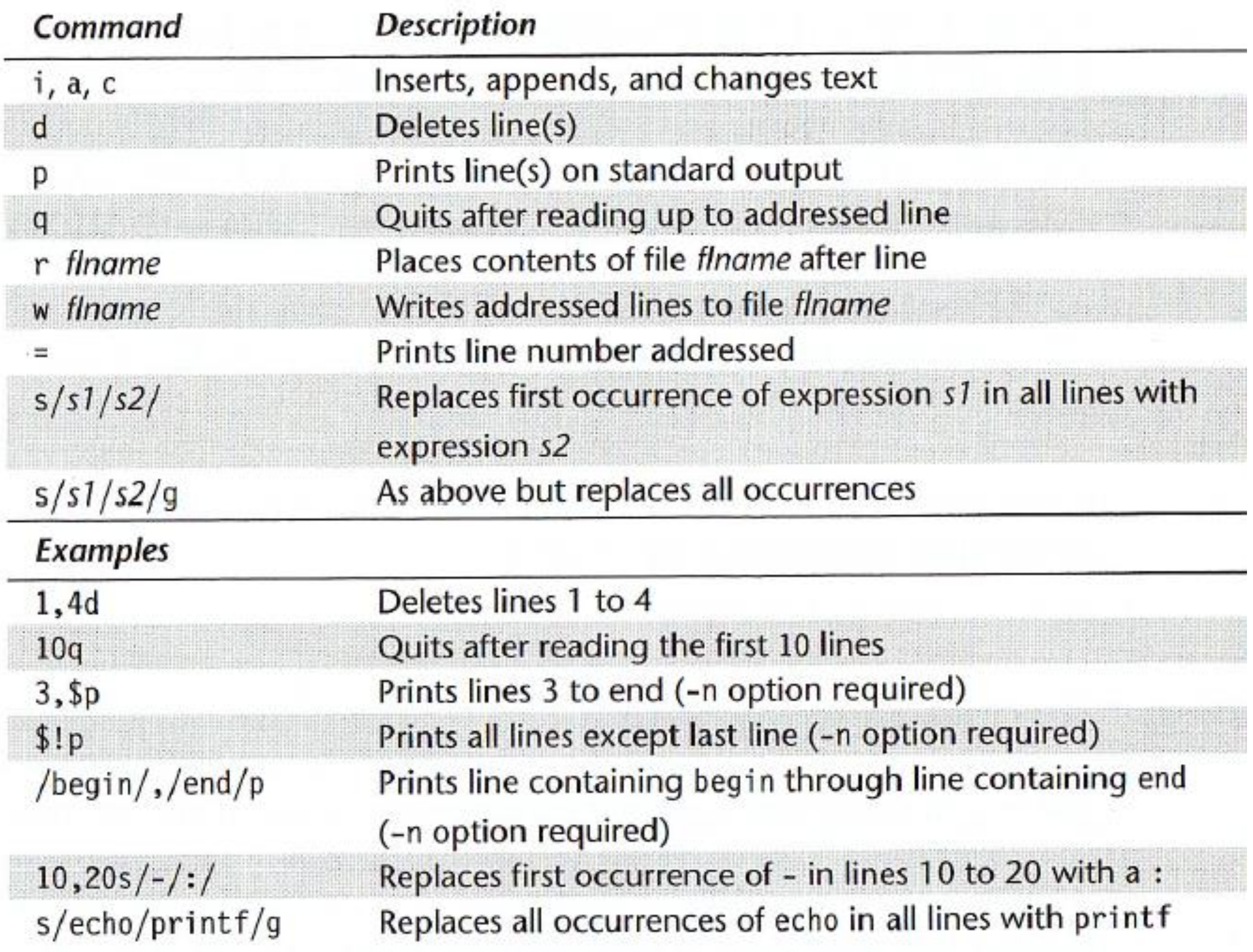
```
sed ‘s/cs390/cs590/g’ syllabus.txt
```

- single or double quotes for sed commands
- `-n`: suppresses automatic printing of pattern space
- `-e`: for more than one editing commands

```
  sed -e ‘1,3d’ -e ‘s/Hemenway/Jones/’ datafile ⇔
  sed ‘1,3d; s/Hemenway/Jones/’ datafile
```

- Get help online
  - `sed -h`
  - `man sed`

Table 10.4 (p278)



Command	*Description*
`i, a, c`	Inserts, appends, and changes text
`d`	Deletes line(s)
`p`	Prints line(s) on standard output
`q`	Quits after reading up to addressed line
`r` *flname*	Places contents of file *flname* after line
`w` *flname*	Writes addressed lines to file *flname*
`=`	Prints line number addressed
`s/`*s1*`/`*s2*`/`	Replaces first occurrence of expression *s1* in all lines with expression *s2*
`s/`*s1*`/`*s2*`/g`	As above but replaces all occurrences

Examples	
`1,4d`	Deletes lines 1 to 4
`10q`	Quits after reading the first 10 lines
`3,$p`	Prints lines 3 to end (-n option required)
`$!p`	Prints all lines except last line (-n option required)
`/begin/,/end/p`	Prints line containing `begin` through line containing `end` (-n option required)
`10,20s/-/:/`	Replaces first occurrence of - in lines 10 to 20 with a :
`s/echo/printf/g`	Replaces all occurrences of `echo` in all lines with `printf`

SED Commands with “pattern”

- **`sed  '/north/p' datafile.txt`**
  - prints both the original and processed lines to the screen
- **`sed -n '/north/p' datafile.txt`**
  - Prints only the lines containing “north”.
- **`sed -n '/north/d' textfile`** (No output)
- **`sed '/north/d' textfile`**
  - Prints the rest of the file to the screen.
- **`sed 's/west/north/g' textfile`** **`(why g?)`**
  - Prints the modified text file to the screen, the original text file is NOT altered.
- **`sed -n 's/west/north/g' textfile`**
  - Any output?
- **`sed -n 's/Hemenway/Jones/gp' datafile`**
- **`sed -n 's/^west/north/p' datafile`**
- **`sed '/north/q' textfile`**
  - Print the lines up to the line containing **`north`**, then quit/exit

sed Command with Addressing

Specify line range to “edit”.

- Line Addressing: Using Line number
  - `sed ‘1,3p’ textfile vs. sed –n ‘1,3p’ textfile`
  - `sed ‘1,3d’ textfile vs. sed –n ‘1,3d’ textfile`
  - `sed ‘3,$d’ textfile`
  - `sed ‘$d’ textfile; sed –n ‘$p’ textfile`
  - `sed ‘3d’ textfile`
  - `Sed “3q” textfile` ← → `head –3 textfile`
- Context (pattern) addressing
  - `sed –n ‘/west/, /east/p’ datafile`
  - `sed ‘/report/s/north/south/g’ datafile`
- Line number and context combination
  - `sed –n ‘5, /^northeast/p’ datafile`

  Prints content from 5th line to the first occurrence of the line starting with northeast

More sed Examples

- "`r`": Reading from files
  - Allows sed to load content from one text file into the current at specified location (after the line with the pattern)
  - `sed '/Suan/r newfile' datafile.txt`
- "`w`": Write matched lines to a file
  - `sed -n '/north/w newfile' datafile.txt`
- "`a`": Append at certain location (insert after the matched line)

```
sed '/^north /aTHE NORTH SALES DISTRICT HAS MOVED'
   datafile.txt
```

- "`i`": Insert above the found lines (insert before the matched line)

```
sed '/eastern/iNEW ENGLAND REGION' datafile
```

- "`c`": Change (Replace) the whole line with new input

```
sed '/eastern/c THE EASTERN REGION HAS BEEN CLOSED'
   datafile
```

- Search and act with multiple commands, using {…} and separate the commands with ";"

```
sed '/Lewis/{ s/Lewis/Joseph/; q; }' datafile.txt ←→
sed -e '/Lewis/s/Lewis/Joseph/' -e /Lewis/q' datafile.txt
sed '/Lewis/s/Lewis/Joseph/;/Lewis/q' datafile.txt
```

"sed" Substitution with Regex

- The Repeated Pattern (&, sed only regex)

  `sed 's/[0-9][0-9]$/&.5/' datafile`

  - &: represents exactly what was found
  - Each line ending with at least two digits will be appended with `.5`

  `sed 's/JONE/***&***/' datafile`

- Tagged Regular Expression(quoted digits)

  `'s/\(henry\) \(Higgins\)/\2,\1)/'`

  `'s/\(Mar\)got/\1ianne/p'` ➔ Margot is replaced with Marianne

- `sed '/west/,/east/s/$/**VACA**/' datafile`,

  From lines containing west to lines containing east were appended (the $ is replaced) with `**VACA**` at the end of line

  - Notice the difference b/w `s/$/**VACA**` and `a **VACA**`

More sed Examples

- `sed -n '/love/p' filename`
  - print all lines containing "love"
- `sed '1,3d' file.txt`
  - lines 1 through 3 are deleted, the rest of lines displayed to screen
- `sed '/Tom/d" file`
  - delete all lines containing the pattern Tom (lines not containing Tom are printed to screen
- `sed '/Tom/!d' file.txt`
  - delete all lines NOT containing the pattern "Tom"
- `sed -n '20,30p' file.txt`
  - print lines of 20 to 30
- `sed  '/^$/d' file.txt`

  delete all empty lines
- `sed  '/^ *$/d' file.txt`
  - delete all blank lines, including empty lines (notice the different from the previous one)
- `sed '5q' datafile` sed quits after printing 5 lines

Edit File with SED

SED will display the edited lines to the screen. It will not change the original file by default

- Two steps when using sed to edit file:
  - Redirecting: `sed ‘1,3d’ file.txt >temp`
  - Renaming: `mv temp file.txt`
- With –i[SUFFIX] option: (--in-place)

  ```
  sed –i.orig ‘1,3d’ file.txt
  ```

  - Changes will be saved in file.txt, the original file is copied to file.txt.orig if suffix is provided

***Warning

Pay attention to when output suppression (-n) should be used

- Don’t use “-n” for deleting actions.
- In other cases, it’s really problem dependent. Always check the output to see if you have got what you want.

sed Scripting

- Use `-f sed.txt` to tell sed to read the commands from a file (sed script)
- A sed script is a list of sed commands in a file
  - # for comments, will not be executed
  - Absolutely NO trailing white space or text at the end of the command
  - Back slash is used for line continuation
  - No quotes are needed in sed script
- How does sed use the script file to process text file?
  1. SED reads in one line from the text file to be processed to the pattern buffer (in sequence)
  2. Then it goes through all the SED commands in the sed script file
  3. After “execute” all the commands in the sed script, it sends the content of pattern buffer (the processed line) to the stdout (if –n is not used)
  4. SED reads in the next line from the text file ….

```
# My first sed script
/Lewis/a\
Lewis is the TOP Salesperson for April!!\
Lewis is moving to the southern district
next month.\
CONGRATULATIONS!
/Margot/c\
*******************\
MARGOT HAS RETIRED\
*******************
1i\
EMPLOYEE DATABASE\
---------------------
$d
```

```
➢sed -f sed1.txt datafile
EMPLOYEE DATABASE
---------------------
northwest   NW   Charles Main      3.0   .98  3   34
western     WE   Sharon Gray       5.3   .97  5   23
southwest   SW   Lewis Dalsass     2.7   .8   2   18
Lewis is the TOP Salesperson for April!!!
Lewis is moving to the southern district next month.
CONGRATULATIONS!
southern    SO   Suan Chin         5.1   .95  4   15
southeast   SE   Patricia Hemenway 4.0   .7   4   17
eastern     EA   TB Savage         4.4   .84  5   20
northeast   NE   AM Main Jr.       5.1   .94  3   13
*******************
MARGOT HAS RETIRED
*******************
```

Pattern Holding Buffer

- There is an additional special buffer (**holding buffer**) for holding the processed line up-to 8192 bytes (?)
  - `h:` send the matching line to the holding buffer
  - `g` or `G:` Retrieving the content from the holding buffer
    - `G:` append content of the holding buffer to the content in space buffer
    - `g:` replace content in space buffer with the content in holding buffer
- Example

  ```
  sed '/northeast/h; $G' datafile
  ```

  - When line containing northeast is read into the pattern (space) buffer, it will be sent to the pattern holding buffer (what inside the buffer previously will be cleared)
  - When the last line is read in, the `G` tells sed to get the content of the holding buffer, append it to the last line in the current pattern buffer
  - What will the above sed command do?
    - Lines containing "northeast" will be copied and appended to the end of file
- Question: In the previous example, if there are more than one line containing "northeast", what are the output?

Holding and Exchanging

- The line can also be exchanged with `x` command
  - `sed` exchanges the content in the holding buffer with what is currently in the pattern buffer
- Example

  ```
  sed "/Patricia/h; /Margot/x' datafile
  ```

  - When line with Margot is found, the content in pattern buffer and holding buffer will be exchanged
  - Result:
    - Line containing Margot will be replaced with the line containing Patricia

***Note:

This is line substitution, different from "pattern substitution"

Holding Buffer Example

Reverse the line order of a text file

```
sed '1!G; h; $!d' num.txt > reversed.txt
```

- sed commands explained
  - '**1!G**': will not do the "G" (get) for the first line
  - '**h**': send the content of pattern space to the holding space (for every line)
  - '**$!d**' delete the content of pattern space if it is NOT the last line (delete every line but the last)
- Execution explained next…

```
# file names.txt
   John
   Smith
   Paul
```

Execution Explained …

- John is put in the pattern space buffer
  - Then is sent to the holding buffer,
  - pattern buffer is deleted (not the last line)
- Smith is read into the pattern buffer,
  - “G” is executed, so John is retrieved from the holding buffer and append to “Smith”, now the pattern space contains two lines: Smith followed with John
  - ‘h’ is executed, so the two lines are sent to the holding buffer
  - “$!d” is executed (the content in pattern buffer is deleted)
- Paul is read into the pattern space buffer
  - “G”, the content of the holding buffer is retrieved and appended after “Paul”; then 3 lines in the order of Paul, Smith, John are in the pattern space buffer
  - The three lines are sent to the holding buffer.
  - Since “Paul” is the last line, the content in the pattern space is NOT deleted, it is displayed to the screen, and then redirected to file `reversed.txt`

Other ways …

- Here is another way to reverse the order of a text file:

  `sed -n '1!G; h; $p' num.txt`

- Using sed script **`sed.txt`**

  `sed -n -f sed.txt names.txt > reversed.txt`

  Here is the script:

```
#This is a sed script for reversing the order of input text file
1!G
h
$p
```

Some Advanced sed Usage

- How to swap two fields, such as first and last name?
  - Text file `file.txt` contains:

    ```
    John Smith is too young.
    Sarah Palin is from Alaska.
    ```

    - What are the output of the following command?

    ```
      sed 's/\(^[^ ]*\) \([^ ]*\)/\2, \1/' file.txt
    vs. sed 's/\(^[^ ].*\) \([^ ].*\)/\2, \1/' file.txt
    ```

  - Text file `file.txt` contains:

    ```
    Smith, John is too young.
    Palin, Sarah is from Alaska.
    ```

    - What are the output of the following command?

    ```
    sed 's/\(^[^ ]*\), \([^ ]*\)/\2 \1/' file.txt
    sed 's/\(^[^,]*\), \([^ ]*\)/\2 \1/' file.txt
    ```

  - Text file `file.txt` contains:

    ```
    John Smith|824-8888.
    Sarah Palin|1-800-Paline.
    ```

    - What are the output of the following command?

    ```
    sed 's/^\([^ ]*\) \([^ ]*\)/\2, \1/' file.txt
    sed 's/^\([^ ]*\) \([^ ]*\)|/\2, \1/' file.txt
    sed 's/^\([^ ]*\) \([^|]*\)/\2, \1/' file.txt
    ```

More…

- Execution Sequence of `sed` Commands
  - File `file.txt` contains the following:

```
Tom is on Monday.
Jerry is on Tuesday.
John is on Wednesday.
```

  - Requirement: reset (cycling-up the dates)
    - `Jerry=>Monday; John=>Tuesday; Tom=>Wednesday`
    - `Tuesday->Monday; Wednesday->Tuesday, Mon->Wed`
  - Can the following command do the replacement so that Tom will be on Wednesday and Jerry will be on Monday?

```
sed 's/Monday/Tuesday/; s/Tuesday/Wednesday/;
  s/Wednesday/Monday/' file.txt
```

  - What's the output of the above command?
  - How to accomplish that change?

Some “sed” Exercises

- Delete all leading and trailing whitespace for each line
- Insert 5 blank spaces at the beginning of each lines
- Substitute “Tuesday” to Wednesday only when the line contains “John”
- Substitute “Tuesday” to Wednesday for lines NOT containing “Smith”
- Print number 10 line only (the 10th line of the file)
- Delete the last line of a text file
- Remove all blank lines of a text file
- Print only lines containing 20 or more characters
- Print lines contains 20 or less characters
- Implement the following sed commands to see the output…
  - `sed -n ‘$=’ file.txt`
  - `sed G myfile.txt > newfile.txt`