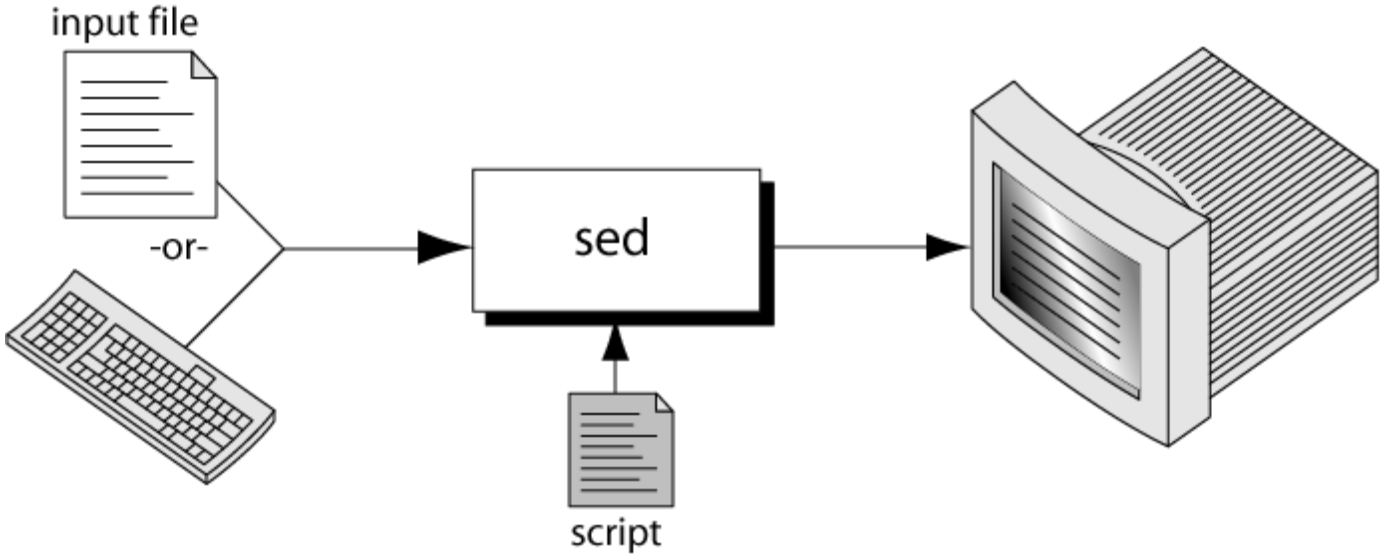
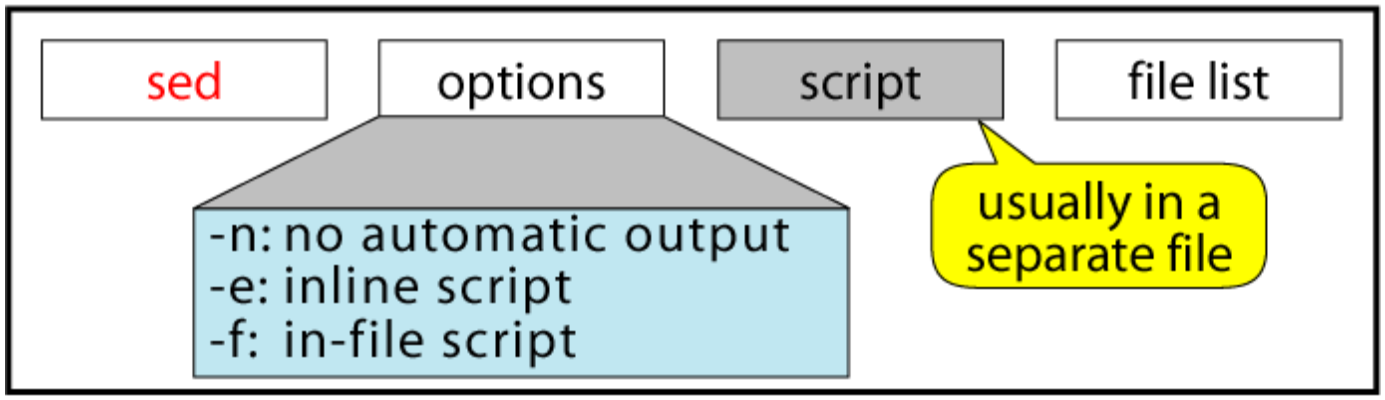
Shell
sed

What is sed?

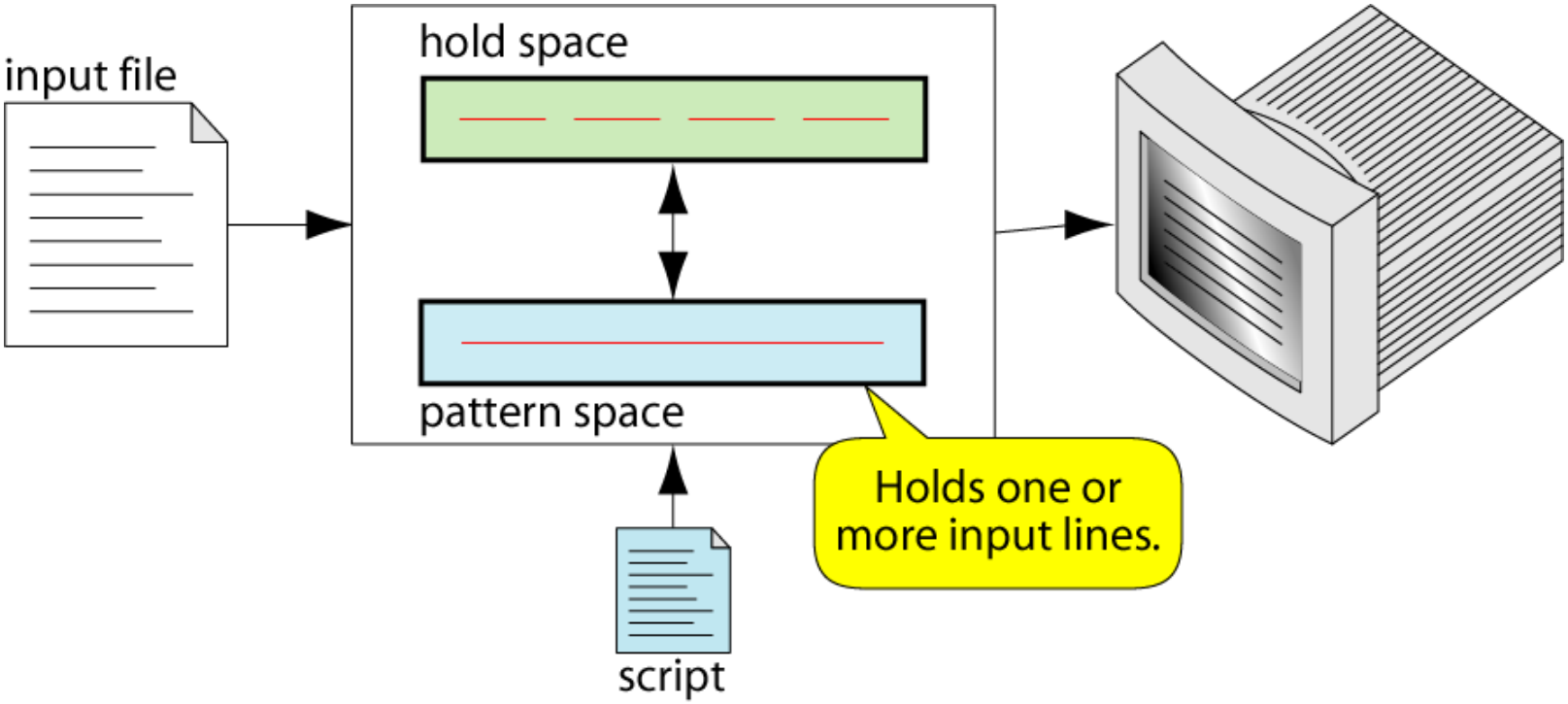
- Sed : a “Stream EDitor”
- A non-interactive stream editor
- Interprets sed instructions and performs actions
- Use sed to:
  - Automatically perform edits on file(s)

The sed command





sed Operation



sed Operation

While (read line){

1) Sed reads an input line from STDIN or a given file, one line at a time, into the *pattern space*.

Pattern Space = a data buffer - the “current text” as it’s being edited

2) For each line, sed executes a series of editing commands (written by the user, you) on the pattern space.

3) Writes the pattern space to STDOUT.

}

Invoking Sed Commands

$ sed [-e script] [-f script-file] [-n] [files…]

-e an "in-line" script, i.e. a script to sed execute given on the command line. Multiple command line scripts can be given, each with an -e option.

-f read scripts from specified file, several -f options can appear

files are the files to read, if a "-" appears, read from stdin, **if no files are given, read also from stdin**

Invoking Sed Commands

-n by default, sed writes each line to stdout when it reaches the end of the script (being whatever on the line) this option prevents that. i.e. no output unless there is a command to order SED specifically to do it

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

Regular Expressions

^ matches the beginning of the line

$ matches the end of the line

. Matches any single character

\ Escapes any metacharacter that follows

(char)* Match arbitrarily many occurences (character)

(char)? Match 0 or 1 instance of (character)

(char)+ Match 1 or more instances of (character)

Regular Expressions (cntd…)

(char)\{m,n\} Match m-n repetitions of (char)

(char)\{m,\} Match m or more repetitions of (char)

(char)\{,n\} Match n or less (possibly 0) repetitions of (char)

(char)\{n\} Match exactly n repetitions of (character)

Regular Expressions (cntd…)

The following **character classes** are short-hand for matching special characters.

[:alnum:]	**Printable characters** (includes white space)
[:alpha:]	Alphabetic characters
[:blank:]	**Space and tab characters**
[:cntrl:]	Control characters
[:digit:]	**Numeric characters**
[:graph:]	Printable and visible (non-space) characters

Regular Expressions (cntd...)

[:lower:]	Lowercase characters
[:print:]	Alphanumeric characters
[:punct:]	**Punctuation characters**
[:space:]	Whitespace characters
[:upper:]	Uppercase characters
[:xdigit:]	Hexadecimal digits

Regular Expressions (cntd…)

/^M.*/ Line begins with capital M, 0 or more chars follow

/..*/ At least 1 character long (/.+/ means the same thing)

/^$/ The empty line

ab|cd Either ‘ab’ or ‘cd’

sed substitution (s)

sed s/day/night/ old >new

echo day | sed s/day/night/
>>night

echo Sunday | sed 's/day/night/'
>>Sunnight

sed substitution (s)

> cat file.txt

unix is great os. unix is opensource. unix is free os.

learn operating system.

unixlinux which one you choose.

sed substitution (s)

>sed 's/unix/linux/2' file.txt

unix is great os. linux is opensource. unix is free os.

learn operating system.

unixlinux which one you choose.

>> ‘g’ or ‘I’ also.

sed substitution (s)

```
>echo "123 abc" | sed 's/[0-9]*/& &/'
  123 123 abc

>echo "123 abc" | sed 's/[0-9][0-9]*/& &/'
  123 123 abc
```

Is sed recursive?

sed 's/loop/loop the loop/g' <old >new

??

This will not cause an infinite loop.

sed print (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

sed print (p)

Here are ways to duplicate the function of *grep* with *sed*:

sed -n 's/pattern/&/p' <file

Or we can use

sed -n '/pattern/p' file

pattern is a RE…

Multiple commands with -e command

sed -e 's/a/A/' -e 's/b/B/' <old >new

Passing arguments into a sed script

```
#!/bin/sh
sed -n 's/'$1'/&/p'
```

Addresses and Ranges of Text

- Restricting to a line number
- The simplest restriction is a line number. If you wanted to delete the first number on line 3, just add a "3" before the command:

```
sed '3 s/[0-9][0-9]*//' <file >new
```

Addresses and Ranges of Text

- Patterns as restriction.

- To delete the **first number** on all lines that start with a "#," use:

```
sed '/^#/ s/[0-9][0-9]*//'
```

Addresses and Ranges of Text

- Ranges by line number
- You can specify a range on line numbers by inserting a comma between the numbers.
- To restrict a substitution to the first 100 lines, you can use:

  sed '1,100 s/A/a/'

Addresses and Ranges of Text

- Ranges by line number (cntd…)
- An easier way is to use the special character "$," which means the last line in the file.
- sed '101,$ s/A/a/'

Addresses and Ranges of Text

- Ranges by patterns
- You can specify two regular expressions as the range.
- Assuming a "#" starts a comment, you can search for a keyword, **remove all comments until you see the second keyword**.
- In this case the two keywords are "start" and "stop:“
- **sed '/start/,/stop/ s/#.*//'**

Delete with d

- You can delete the lines a file by specifying the line number or a range or numbers.

```
>sed '2 d' file.txt
>sed '5,$ d' file.txt
>sed '/^#/ d' file.txt
```

Delete with d

- You can delete the lines a file by specifying the line number or a range or numbers.

```
>sed '2 d' file.txt
>sed '5,$ d' file.txt
>sed '/^#/ d‘ file.txt
```

Reversing the restriction with !

- The "!" character, which often means *not* in UNIX utilities, inverts the address restriction.

sed -n '/pattern/ !p' </tmp/b

--prints all lines that don't contain the pattern

Relationships between d, p, and !

Sed	Range	Command	Printed Results
sed -n	1,10	p	first 10 lines
sed -n	11,$	!p	first 10 lines
sed	1,10	!d	first 10 lines
sed	11,$	d	first 10 lines

The q or quit command

- This command is most useful when you wish to abort the editing after some condition is reached.

sed '11 q'

- which quits when the eleventh line is reached.

The q or quit command

- Display the first 50 lines and quit

$ sed -e ’50q’ datafile

>>>> ?? sed '1,10 q'

Append a line with 'a'

- The "a" command appends a line after the range or pattern.

```
#!/bin/sh
sed '/WORD/ a\
Add this line after every line with WORD'
```

Append a line with 'a'

- Adding more than one line

```
#!/bin/sh
sed '
/WORD/ a\
Add this line\
This line\
And this line
'
```

Insert a line with 'i'

- You can insert a new line before the pattern with the "i" command:

```
#!/bin/sh
sed ' /WORD/ i\
Add this line before every line with WORD'
```

Change a line with 'c'

- You can change the current line with a new line.

```
#!/bin/sh
sed ' /WORD/ c\
Replace the current line with the line'
```

Print line number with =

- The "=" command prints the current line number to standard output.
- One way to find out the line numbers that contain a pattern is to use:

```
sed -n '/PATTERN/ =' file
```

Transform with y

- The sed command can be used to convert the lower case letters to upper case letters by using the transform "y" option.

  For instance, to change the letters "a" through "f" into their upper case form, use:

  sed 'y/abcdef/ABCDEF/' file

Combined examples

> cat file.txt

unix is great os. unix is opensource. unix is free os.

learn operating system.

unixlinux which one you choose.

Combined examples

> sed 's/unix/{&&}/' file.txt

{unixunix} is great os. unix is opensource. unix is free os.

learn operating system.

{unixunix}linux which one you choose.

Combined examples

>sed 's/unix/linux/p' file.txt

linux is great os. unix is opensource. unix is free os.

linux is great os. unix is opensource. unix is free os. learn operating system.

linuxlinux which one you choose.

linuxlinux which one you choose.

Combined examples

- >sed -n 's/unix/linux/p' file.txt

  linux is great os. unix is opensource. unix is free os.

  linuxlinux which one you choose.

Combined examples

>sed '/linux/ s/unix/centos/' file.txt

unix is great os. unix is opensource. unix is free os.

learn operating system.

centoslinux which one you choose.

Combined examples

>sed '/unix/ a "Add a new line"' file.txt

unix is great os. unix is opensource. unix is free os.

"Add a new line"

learn operating system. unixlinux which one you choose.

"Add a new line"

Combined examples

>sed '/unix/ i "Add a new line"' file.txt

"Add a new line"

unix is great os. unix is opensource. unix is free os.

learn operating system.

"Add a new line"

unixlinux which one you choose.

Combined examples

>sed '/unix/ c "Change line"' file.txt

"Change line"

learn operating system.

"Change line"

Combined examples

>sed 'y/ul/UL/' file.txt

Unix is great os. Unix is opensoUrce. Unix is free os.

Learn operating system.

UnixLinUx which one yoU choose.

The slash as a delimiter

- If you want to change a pathname that contains a slash –

  say **/usr/local/bin** to **/common/bin**

- you could use the backslash to quote the slash:
- sed 's/\/usr\/local\/bin/\/common\/bin/' <old >new

Writing a file with the 'w' 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

Writing a file with the 'w' command

- Example that will only write lines that start with an even number

```
sed -n '/^[0-9]*[02468]/ w even' <file
```