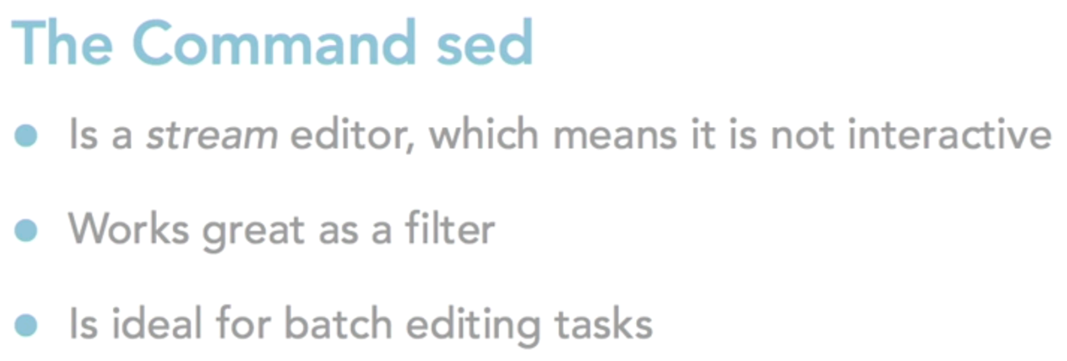
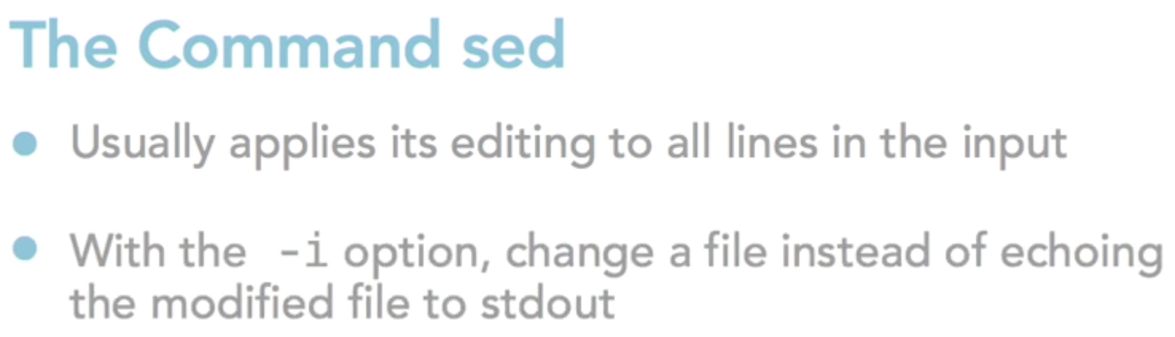
SED AND AWK

Tuesday, 18 May 2021 4:50 PM



The Command sed

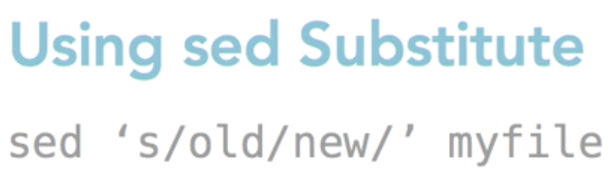
- Is a *stream* editor, which means it is not interactive
- Works great as a filter
- Is ideal for batch editing tasks



The Command sed

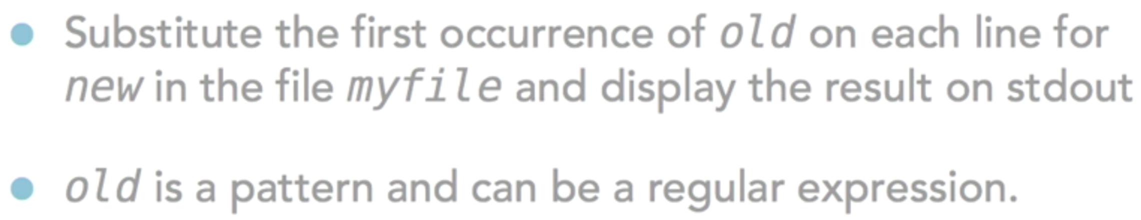
- Usually applies its editing to all lines in the input
- With the `-i` option, change a file instead of echoing the modified file to stdout

USING SED's SUBSTITUTE COMMAND:

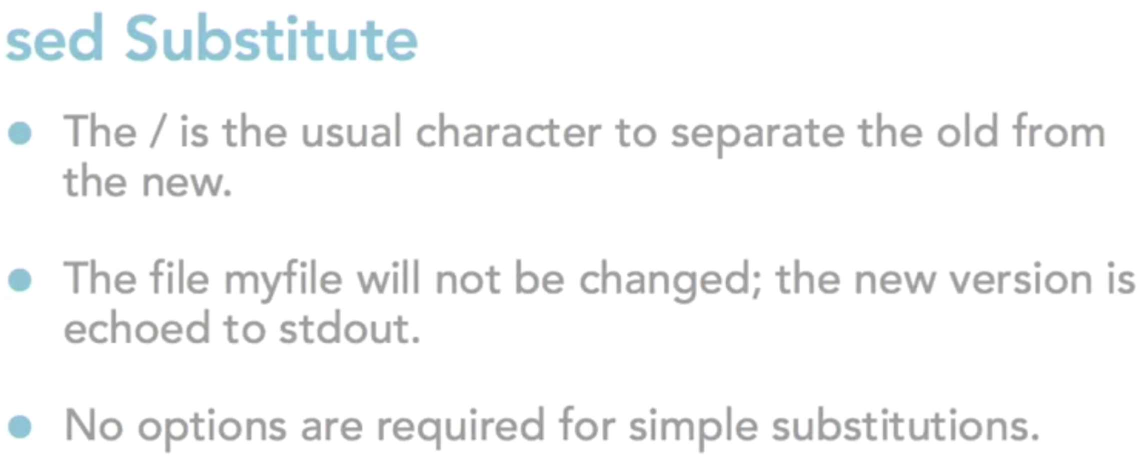


Using sed Substitute

```
sed ‘s/old/new/’ myfile
```



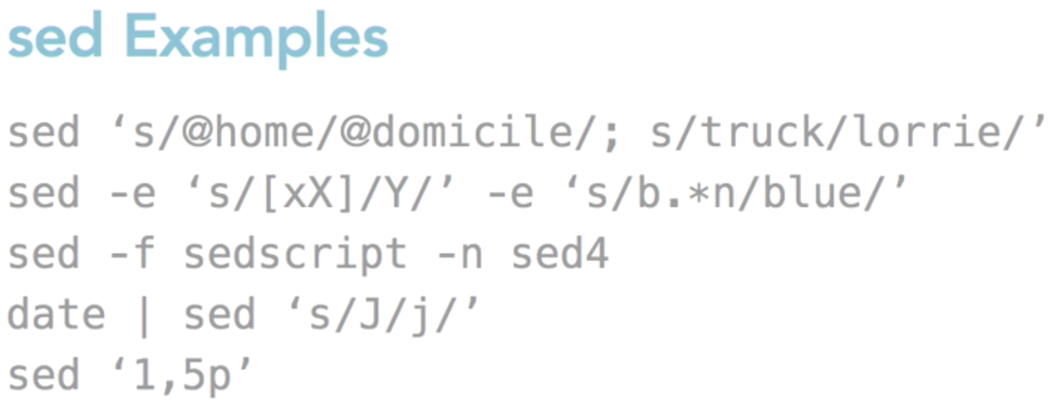
- Substitute the first occurrence of *`old`* on each line for *`new`* in the file *`myfile`* and display the result on stdout
- *`old`* is a pattern and can be a regular expression.



sed Substitute

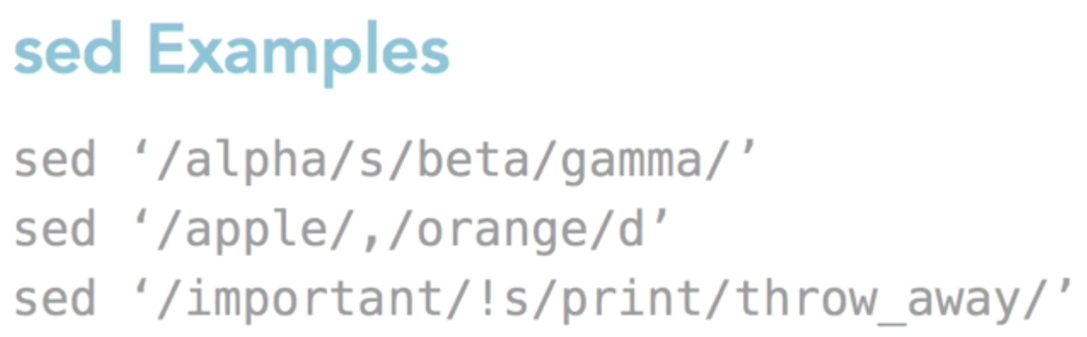
- The / is the usual character to separate the old from the new.
- The file myfile will not be changed; the new version is echoed to stdout.
- No options are required for simple substitutions.

EXAMPLES OF USING SED:



sed Examples

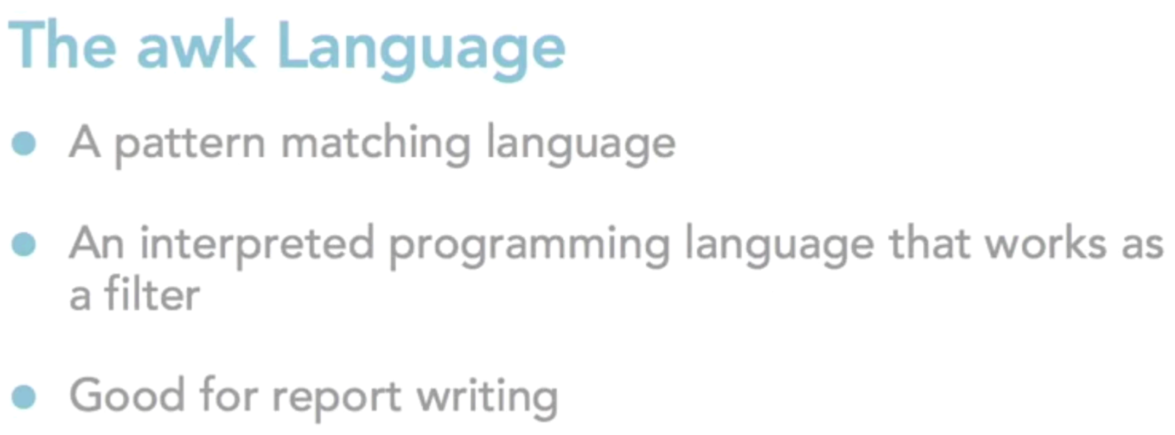
```
sed ‘s/@home/@domicile/; s/truck/lorrie/’
sed -e ‘s/[xX]/Y/’ -e ‘s/b.*n/blue/’
sed -f sedscript -n sed4
date | sed ‘s/J/j/’
sed ‘1,5p’
```



sed Examples

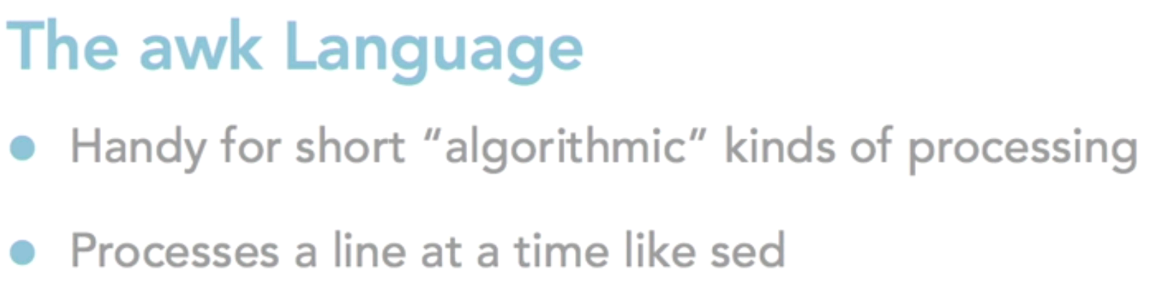
```
sed ‘/alpha/s/beta/gamma/’
sed ‘/apple/,/orange/d’
sed ‘/important/!s/print/throw_away/’
```

AWK:



The awk Language

- A pattern matching language
- An interpreted programming language that works as a filter
- Good for report writing



The awk Language

- Handy for short “algorithmic” kinds of processing
- Processes a line at a time like sed

- Breaks each line into fields, $1, $2, etc.

The awk Language

- Fields are delimited by the values in the variable FS, normally white space.
- `$0` is the entire line (record).