COMP2041/9044

Week 2 - Filtering

The core 7 filters

- sort
- wc
- tr
- head / tail
- cut
- uniq
- sed

sort

- Sorts your input in **ascending line order**
- Lines can be sorted by a specific character within a field

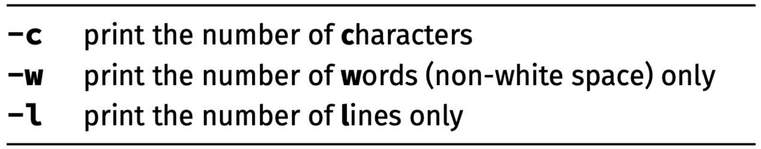
-r	sort in descending order (**r**everse sort)
-n	sort numerically rather than lexicographically
-d	dictionary order: ignore non-letters and non-digits
-tc	use character c to separate columns (default: non-blank to blank transition)
-kn	sort on column n

```
sort -r <filename>
"tab"
1
10
11
12
13

zid|name|phone number

sort -k3 -t"|" < file
```

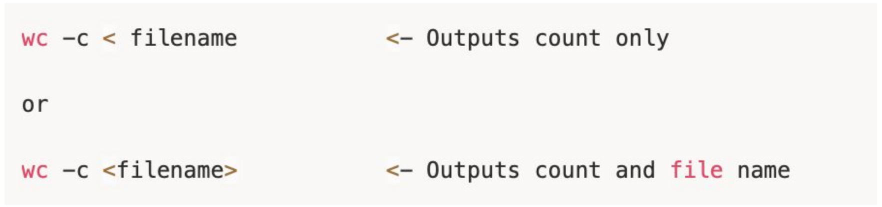
wc

(word count)

- Summarises information about the input



-c	print the number of **c**haracters
-w	print the number of **w**ords (non-white space) only
-l	print the number of **l**ines only

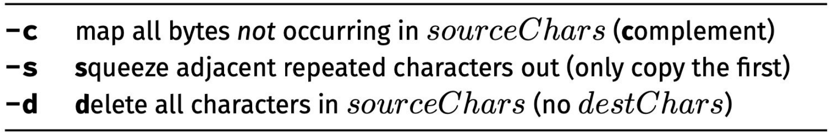


```
wc -c < filename              <- Outputs count only

or

wc -c <filename>              <- Outputs count and file name
```

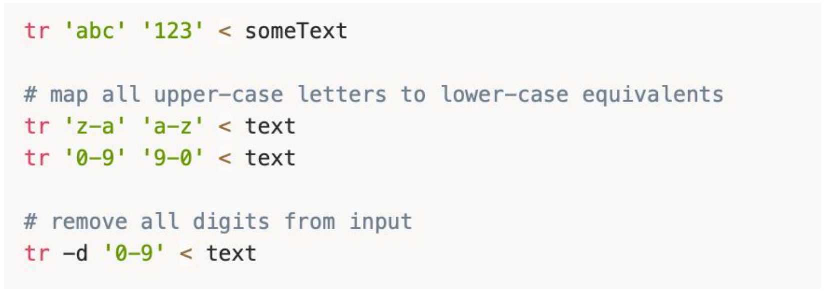
tr

(transliterate)

- Replaces a character with another one
  - Can be a range of characters



-c	map all bytes *not* occurring in $sourceChars$ (**c**omplement)
-s	**s**queeze adjacent repeated characters out (only copy the first)
-d	**d**elete all characters in $sourceChars$ (no $destChars$)



```
tr 'abc' '123' < someText

# map all upper-case letters to lower-case equivalents
tr 'z-a' 'a-z' < text
tr '0-9' '9-0' < text

# remove all digits from input
tr -d '0-9' < text
```

head / tail

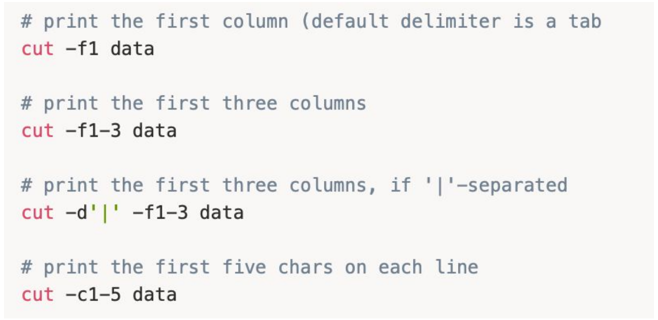
- Selects the first / last **n** lines

```
head -n 30 filename          <-- Prints the first 30 lines of file

tail -n 30 filename          <-- Prints the last 30 lines of file
```

cut

- Selects specific sections of an input using a “**delimiter**”.
    - Input should be in a certain format
    - Example: “field1|field2|field3”

-f$listOfCols$	print only the specified fields (tab-separated) on output
-c$listOfPos$	print only chars in the specified positions
-dc	use character c as the field separator

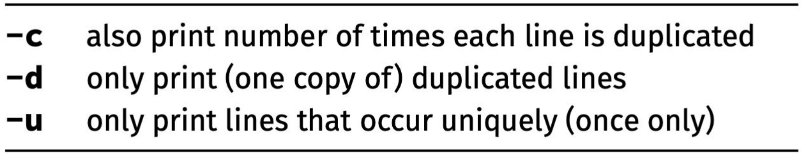


```
# print the first column (default delimiter is a tab
cut -f1 data

# print the first three columns
cut -f1-3 data

# print the first three columns, if '|'-separated
cut -d'|' -f1-3 data

# print the first five chars on each line
cut -c1-5 data
```

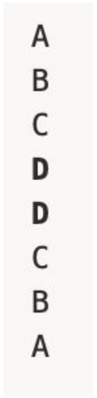
uniq

(unique)

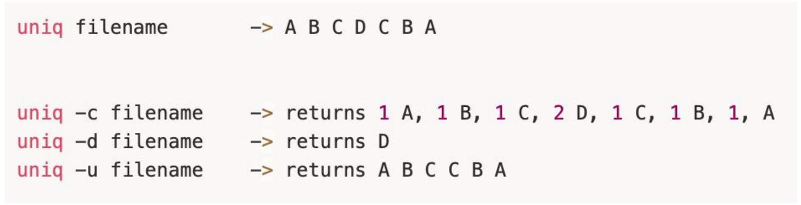
- Removes all **adjacent** lines that are not unique



-c	also print number of times each line is duplicated
-d	only print (one copy of) duplicated lines
-u	only print lines that occur uniquely (once only)



```
A
B
C
D
D
C
B
A
```



```
uniq filename       -> A B C D C B A


uniq -c filename    -> returns 1 A, 1 B, 1 C, 2 D, 1 C, 1 B, 1, A
uniq -d filename    -> returns D
uniq -u filename    -> returns A B C C B A
```

sed

(steam editor)

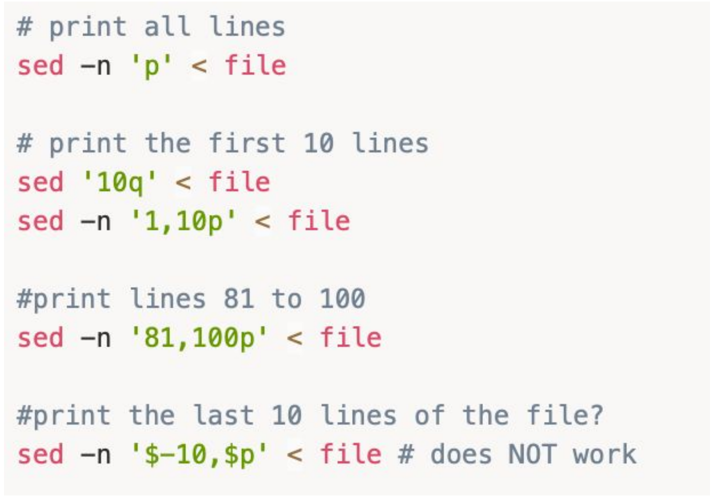
- The most complicated filter you'll see this term 😱

-n	do not print lines by default - applies all editing commands as normal but displays no output, unless p appended to edit command
-E	extended regular expressions like grep you often want this

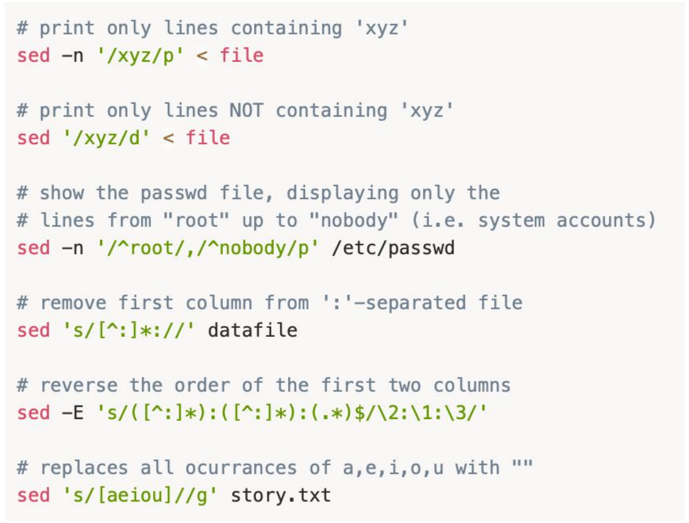
p	print the current line
d	delete (don't print) the current line
s/_regex_**/**_replace_**/**	substitute first occurrence of string matching ***regex*** by ***replace*** string
s/_regex_**/**_replace_**/g**	substitute all occurrences of string matching ***regex*** by ***replace*** string
q	terminate execution of sed

_line_number_	selects the specified line
_start_line_number_**,**_end_line_number_	selects all lines between specified line numbers
/_regex_**/**	selects all lines that match ***regex***
/_regex1_**/,/**_regex2_**/**	selects all lines between lines matching regex1 and regex2

sed examples (from lectures)



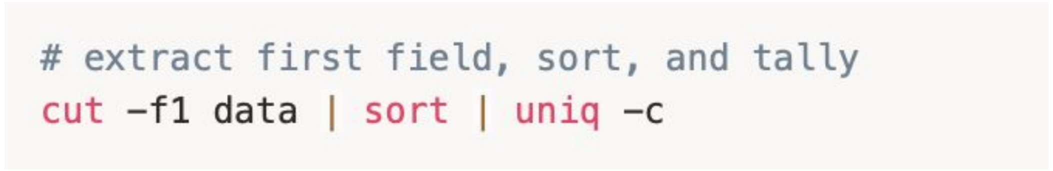
```
# print all lines
sed -n 'p' < file

# print the first 10 lines
sed '10q' < file
sed -n '1,10p' < file

#print lines 81 to 100
sed -n '81,100p' < file

#print the last 10 lines of the file?
sed -n '$-10,$p' < file # does NOT work
```



```
# print only lines containing 'xyz'
sed -n '/xyz/p' < file

# print only lines NOT containing 'xyz'
sed '/xyz/d' < file

# show the passwd file, displaying only the
# lines from "root" up to "nobody" (i.e. system accounts)
sed -n '/^root/,/^nobody/p' /etc/passwd

# remove first column from ':'-separated file
sed 's/[^:]*://' datafile

# reverse the order of the first two columns
sed -E 's/([^:]*):([^:]*):(.*)$/\2:\1:\3/'

# replaces all ocurrances of a,e,i,o,u with ""
sed 's/[aeiou]//g' story.txt
```

Combine multiple filters

Using “|” aka the pipeline



```
# extract first field, sort, and tally
cut -f1 data | sort | uniq -c
```

Tutorial Questions

Question 2-4, 6

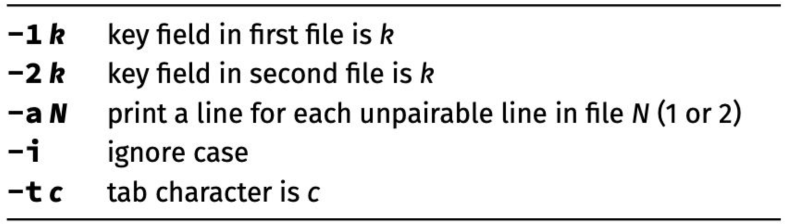
More filters

The less commonly used filters

- join
- find

join

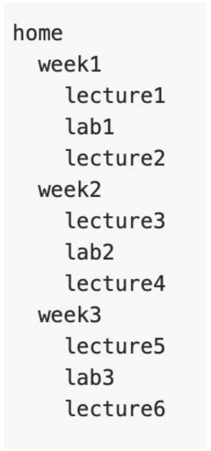
- Allows you to merge lines from two files based on a common field, effectively combining related data.



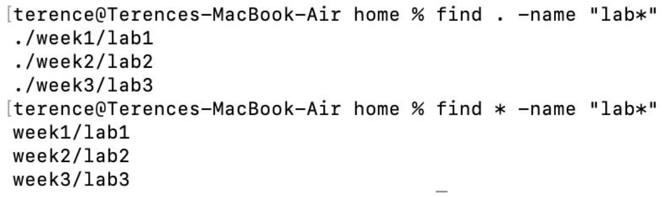
-1 ***k***	key field in first file is *k*
-2 ***k***	key field in second file is *k*
-a ***N***	print a line for each unpairable line in file *N* (1 or 2)
-i	ignore case
-t ***c***	tab character is *c*

find

- Find a file in your directory using regex
- More commonly used than using:
  - “ls | grep -E <regex>”
- Refer to manual to learn more about this



```
home
  week1
    lecture1
    lab1
    lecture2
  week2
    lecture3
    lab2
    lecture4
  week3
    lecture5
    lab3
    lecture6
```



```
[terence@Terences-MacBook-Air home % find . -name "lab*"
./week1/lab1
./week2/lab2
./week3/lab3
[terence@Terences-MacBook-Air home % find * -name "lab*"
week1/lab1
week2/lab2
week3/lab3
```

“.” current directory

“*” expands to a list of everything in folder (not advised)

Regular Expression Resources

Cheat Sheet: (Search “regex cheat sheet” and find the one by David Child)

https://cheatography.com/davechild/cheat-sheets/regular-expressions/

Online Regex Tester:

https://regex101.com/

❗Disclaimer: This website uses a different method to run regular expressions, so there will be some cases where the output differs

Tutorial “Slides”

https://2041.terencelim.dev/26t1-w09a (will be uploaded during the lab)

LAB Time!

Room: Ainsworth 302 (String Lab)