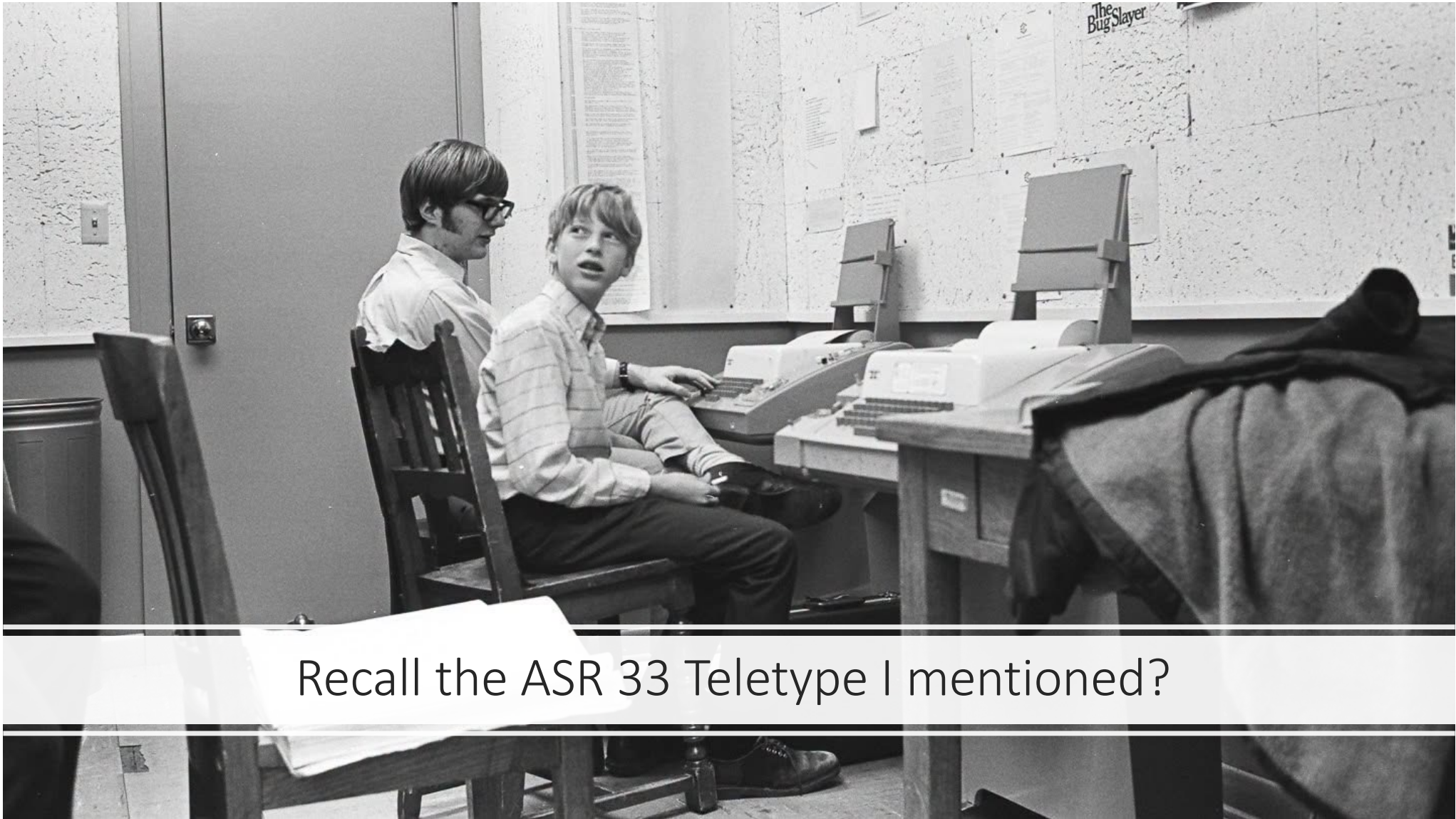


Lecture 6

408/508 Computational Techniques for Linguists

Today's Topics

- the cat (*concatenate & print*) command
- **Shell scripting**
 - *writing a program using an editor and running it!*
- Homework 3 (**due Sunday midnight**)
 - *Step-by-step Bash shell exercises*



Recall the ASR 33 Teletype I mentioned?

cat command

- See <http://www.linfo.org/cat.html>

cat is one of the most frequently used **commands** on **Unix-like operating systems**. It has three related functions with regard to text files: displaying them, combining copies of them and creating new ones.

- Let's use **nano** (text editor to create three files from scratch)
 - file1 – *put more than one screenfull of text in this one*
 - file2
 - file3

cat command

- An exercise:

1. `cat file1` (print contents of file1)
2. `cat file1 | more` ('|' = pipe output to pagination command `more` or `less`)
3. `more file1` (nowadays aliased to `less`)
4. `less file1` (stops at screen bottom, *space* or *f* to show more, *b* up)
5. `cat > file1` (create file1, input from terminal until **Control-D** EOF)
6. `cat` (**weird!** input from terminal goes to terminal)
7. `cat >> file1` (append input from terminal to file file1)
8. `cat file1 > file2` (file copy)
9. `cp file1 file2` – *easier* (`cp` = copy)
10. `cat file1 file2 file3` (prints file[1–3] in order)
11. `cat file[1–3] > file4` (prints all 3 files to file4)
12. `cat file[1–3] | sort > file4` (3 files sorted by ASCII to file4)
13. `cat - file5 > file6` ('-' = input from terminal)
14. `cat file7 - > file8`

Shell program

`{1..10..2}` means range from 1 to 10 incrementing by 2
; (semicolon) OR newline terminates/separates statements

```
2022 Ubuntu [Running]
Activities Terminal Sep 2 15:41
sandikway@sandikway-VirtualBox: ~
~$ nano ~/.bashrc
~$ echo $SHELL
/bin/bash
~$ for i in {1..10}
> do
> echo "$i time through the loop"
> done
1 time through the loop
2 time through the loop
3 time through the loop
4 time through the loop
5 time through the loop
6 time through the loop
7 time through the loop
8 time through the loop
9 time through the loop
10 time through the loop
~$
```

echo means
print

```
2022 Ubuntu
Activities Terminal S
sandikway@sandikway-Vi
~$ for i in {1..10..2}; do echo "$i"; done
1
3
5
7
9
~$
```

not in
macOS

```
~$ for ((i=1; i<=10; i=i+2)); do echo "$i by 2"; done
1 by 2
3 by 2
5 by 2
7 by 2
9 by 2
```

macOS

Input

- At a terminal:
 - `read -p "Name: " name`
 - `read -p "Enter X and Y: " x y`
 - `echo $x`
 - `echo $y`

Shell script

- Use nano to create a new file named `script.sh` (convention `.sh` for script filetype):
 - `nano script.sh`

- Enter:

```
GNU nano 6.2 script.sh
#!/bin/bash
read -p "First number: " x
read -p "Second number: " y
((z= x+y))
((xy = x*y))
echo "$x + $y = $z, $x * $y = $xy"
```

Without chmod!

- Run it!

```
~$ nano script.sh
~$ chmod u+x script.sh
~$ ./script.sh
First number: 5
Second number: 13
5 + 13 = 18, 5 * 13 = 65
~$
```

```
[(base) 508$ ./script.sh
-bash: ./script.sh: Permission denied
(base) 508$
```

- `./script.sh` means run the script in file `script.sh` in the current directory (`.`)
- `chmod u+x filename` means add (+) execute (x) permission for user (u) to *filename*

Comparison operators: `if-then`

- Format:

if [`$x` **OP** `$y`]; then

...

(else/elif...)

fi

- [....] is known as *test* (*need spaces*)

- **OP** (numeric):

- `-eq` *equals*
- `-ne` *not equals*
- `-gt` *greater than*
- `-ge` *greater than or equals*
- `-lt` *less than*
- `-le` *less than or equals*

The `test` command and its aliases return an exit status:

- `0` (success) if the expression evaluates to true.
- `1` (failure) if the expression evaluates to false.

- Examples:

- `echo $x $i`

2 5

- `test $x -le $i`

- `echo $? (exit status)`

0

- `test $x -le $i -a $i -lt $x`

- `echo $?`

1

Shell script 2

- using the same approach as in the first shell script ...

```
GNU nano 6.2 script2.sh
#!/bin/bash
read -p "Number 1: " x
read -p "Number 2: " y
if [ $x -lt $y ]; then
    echo "$x smaller than $y"
elif [ $x -eq $y ]; then
    echo "$x = $y"
else
    echo "$x bigger than $y"
fi
```

- Beware of execute permissions!

```
~$ ./script3.sh
bash: ./script3.sh: Permission denied
~$ chmod u+x script3.sh
~$
```

```
~$ nano script2.sh
~$ ./script2.sh
Number 1: 3
Number 2: 7
3 smaller than 7
~$ ./script2.sh
Number 1: 4
Number 2: 4
4 = 4
~$ ./script2.sh
Number 1: 5
Number 2: 1
5 bigger than 1
~$
```

Shell script 2

Compare these two files:

- newlines sometimes matter
- ; separator must be used in some cases
 - if condition; then
- vs.
 - if *condition*
 - then
- see also ; fi

GNU nano 6.2	script3.sh
<pre>#!/bin/bash read -p "Number 1: " x read -p "Number 2: " y if [\$x -lt \$y]; then echo "\$x smaller than \$y" elif [\$x -eq \$y]; then echo "\$x = \$y" else echo "\$x bigger than \$y"; fi</pre>	
<pre>GNU nano 6.2 script2.sh #!/bin/bash read -p "Number 1: " x read -p "Number 2: " y if [\$x -lt \$y]; then echo "\$x smaller than \$y" elif [\$x -eq \$y]; then echo "\$x = \$y" else echo "\$x bigger than \$y" fi</pre>	

exit value: if-then

- <https://tldp.org/LDP/abs/html/testconstructs.html>
- Examples:

```
if cmp a b &> /dev/null # &> /dev/null means suppress output
then echo "Files a and b are identical."
else echo "Files a and b differ."
fi
```

```
$ cmp file1 file2
file1 file2 differ: char 1, line 1
$ cmp file1 file2 &> /dev/null
$ echo $?
1
```



Key features and usage of cmp :

- **Syntax:** `cmp [OPTION]... FILE1 [FILE2]`
- **Comparison Method:** Performs a byte-by-byte comparison.
- **Output for Identical Files:** If the files are identical, `cmp` produces no output and returns an exit status of `0`.
- **Output for Different Files:** If the files differ, `cmp` reports the byte number and line number where the first difference occurs. It returns an exit status of `1`. 

exit value: `if-then`

The very useful "if-grep" construct:

```
if grep -q Bash file
then echo "File contains at least one occurrence of Bash."
fi
```

- **Pattern Matching:** `grep` searches for lines within specified files (or standard input) that contain a match to a given pattern. This pattern can be a simple string or a more complex regular expression.
- **Output:** By default, `grep` prints the lines that contain a match to standard output.
- **Options:** `grep` offers various options to modify its behavior, such as:
 - `-i` : Ignore case sensitivity during the search.
 - `-r` or `-R` : Recursively search through directories and subdirectories.
 - `-v` : Invert the match, printing lines that do not contain the pattern.
 - `-n` : Display the line number along with the matching line.
 - `-c` : Count the number of matching lines instead of printing them.

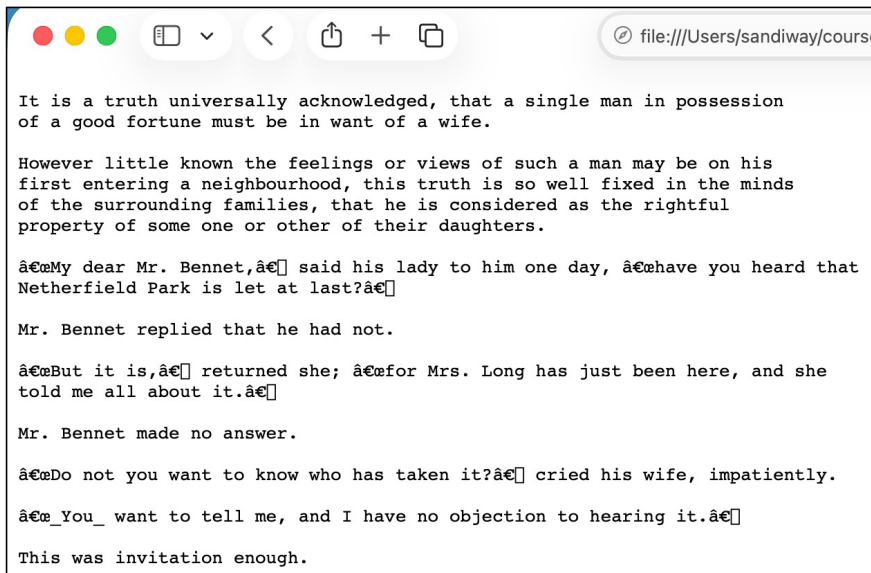
exit value: `if-then`

```
word=Linux
letter_sequence=inu
if echo "$word" | grep -q "$letter_sequence" # "-q" suppresses
output.
then
    echo "$letter_sequence found in $word"
else
    echo "$letter_sequence not found in $word"
fi
```

Homework 3: Exercise 1

1. Download file text.txt from the course website

- You can use the browser (copy) and save the file as **plain text**



A screenshot of a web browser window. The address bar shows a file path: file:///Users/sandway/course. The text content is displayed with a default encoding, resulting in some characters being escaped as \u0026quot;.

It is a truth universally acknowledged, that a single man in possession of a good fortune must be in want of a wife.

However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered as the rightful property of some one or other of their daughters.

\u0026quot;My dear Mr. Bennet,\u0026quot; said his lady to him one day, \u0026quot;have you heard that Netherfield Park is let at last?\u0026quot;

Mr. Bennet replied that he had not.

\u0026quot;But it is,\u0026quot; returned she; \u0026quot;for Mrs. Long has just been here, and she told me all about it.\u0026quot;

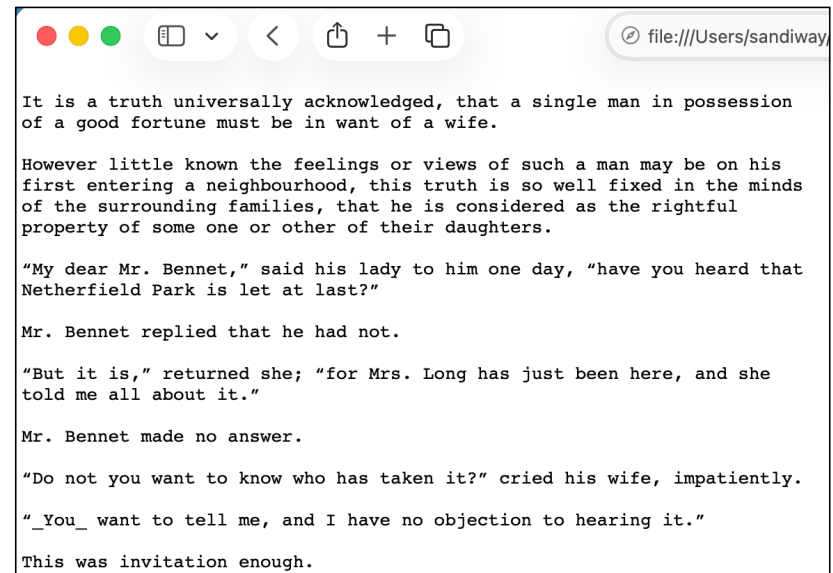
Mr. Bennet made no answer.

\u0026quot;Do not you want to know who has taken it?\u0026quot; cried his wife, impatiently.

\u0026quot;_You_ want to tell me, and I have no objection to hearing it.\u0026quot;

This was invitation enough.

Default encoding: Safari



A screenshot of a web browser window. The address bar shows a file path: file:///Users/sandway/course. The text content is displayed with utf-8 encoding, showing the text with proper quotation marks.

It is a truth universally acknowledged, that a single man in possession of a good fortune must be in want of a wife.

However little known the feelings or views of such a man may be on his first entering a neighbourhood, this truth is so well fixed in the minds of the surrounding families, that he is considered as the rightful property of some one or other of their daughters.

\u0026quot;My dear Mr. Bennet,\u0026quot; said his lady to him one day, \u0026quot;have you heard that Netherfield Park is let at last?\u0026quot;

Mr. Bennet replied that he had not.

\u0026quot;But it is,\u0026quot; returned she; \u0026quot;for Mrs. Long has just been here, and she told me all about it.\u0026quot;

Mr. Bennet made no answer.

\u0026quot;Do not you want to know who has taken it?\u0026quot; cried his wife, impatiently.

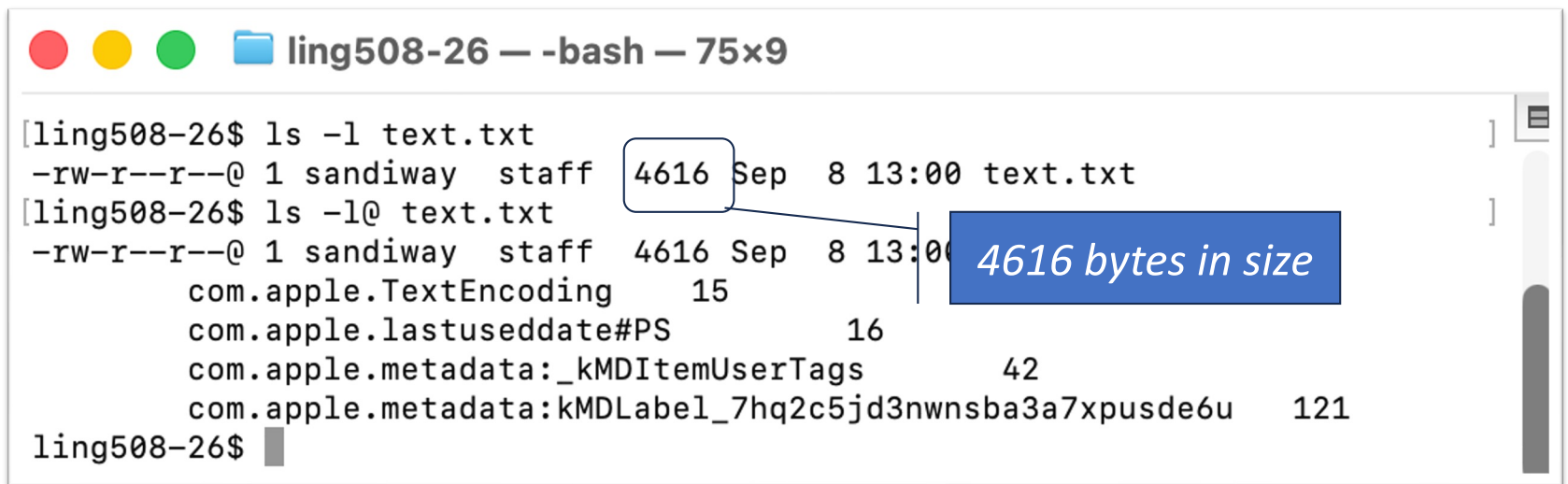
\u0026quot;_You_ want to tell me, and I have no objection to hearing it.\u0026quot;

This was invitation enough.

utf-8: Safari

Homework 3: Exercise 1

2. Check to see the file exists in your directory in the **Terminal**

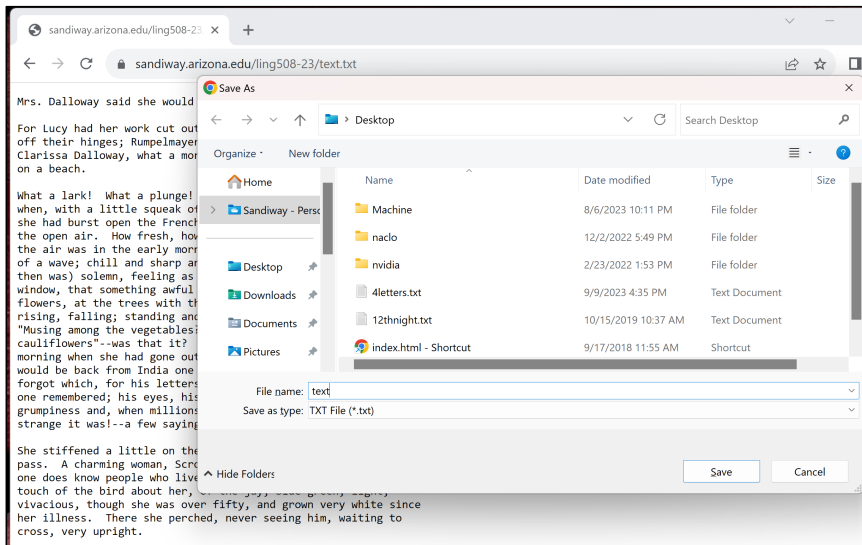


A screenshot of a macOS Terminal window titled "ling508-26 — -bash — 75x9". The window shows the output of two commands. The first command, `ls -l text.txt`, shows the file's permissions, owner, group, size, date, and name. The size "4616" is highlighted with a blue box. The second command, `ls -l@ text.txt`, shows the same information plus extended attributes. A blue box with the text "4616 bytes in size" has an arrow pointing to the "4616" in the second command's output. The terminal text is as follows:

```
[ling508-26$ ls -l text.txt
-rw-r--r--@ 1 sandiway  staff  4616 Sep  8 13:00 text.txt
[ling508-26$ ls -l@ text.txt
-rw-r--r--@ 1 sandiway  staff  4616 Sep  8 13:00
    com.apple.TextEncoding      15
    com.apple.lastuseddate#PS    16
    com.apple.metadata:_kMDItemUserTags 42
    com.apple.metadata:kMDLabel_7hq2c5jd3nwnsba3a7xpusde6u 121
ling508-26$
```

Note: Windows file access from Ubuntu

Lecture 4: can cp (copy)
can access your Windows C: drive
(from within Ubuntu) via directory /mnt/c



```
sandiway@DESKTOP-VEPP64H  x  +  v

System load:  0.13           Processes:           12
Usage of /:   2.8% of 250.92GB Users logged in:       0
Memory usage: 5%           IPv4 address for eth0: 172.27.76.241
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

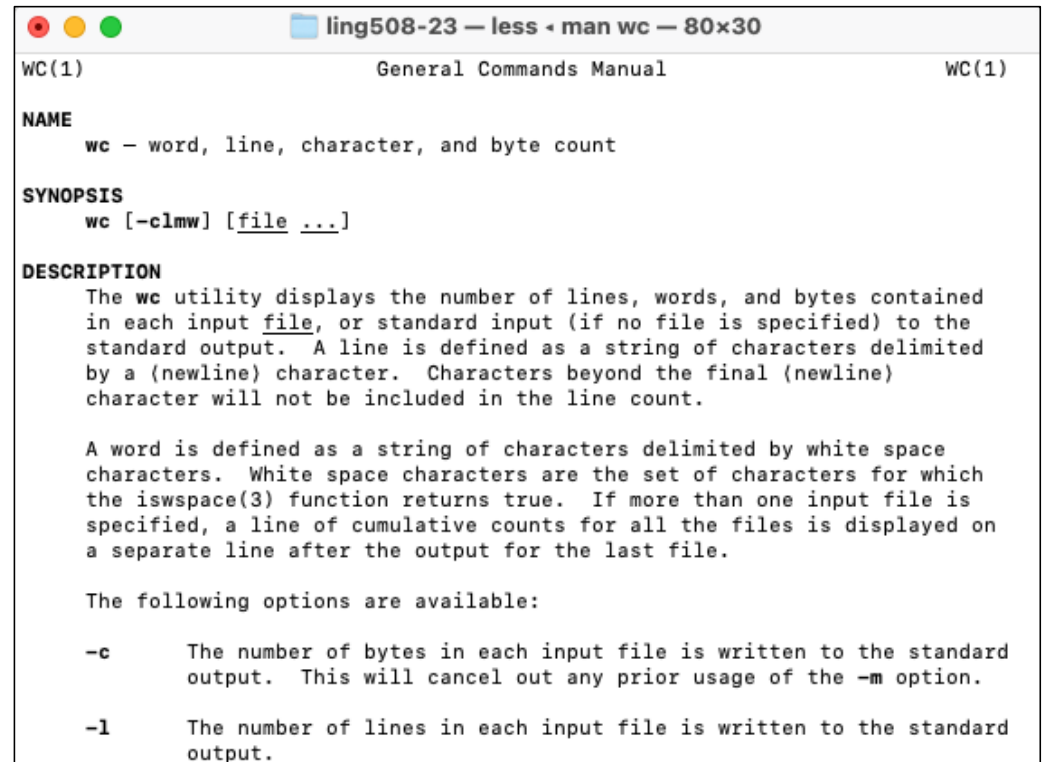
This message is shown once a day. To disable it please create the
/home/sandiway/.hushlogin file.
sandiway@DESKTOP-VEPP64Q:~$ ls /mnt/c/Users/sandiway/Desktop/text.txt
/mnt/c/Users/sandiway/Desktop/text.txt
sandiway@DESKTOP-VEPP64Q:~$ pwd
/home/sandiway
sandiway@DESKTOP-VEPP64Q:~$ cp /mnt/c/Users/sandiway/Desktop/text.txt .
sandiway@DESKTOP-VEPP64Q:~$ ls
0200991.txt  Downloads  cmudict.perl  nltk_data  wordpuzzle2.py
4letters.txt  WordNet-3.0  get-pip.py   text.txt   wordpuzzle6.py
sandiway@DESKTOP-VEPP64Q:~$
```

Homework 3:

Exercise 1

`wc` is a useful command.

3. Do `man wc` to see the manual page (manpage).



```
ling508-23 — less « man wc — 80x30
WC(1)                                     General Commands Manual      WC(1)

NAME
    wc - word, line, character, and byte count

SYNOPSIS
    wc [-clmw] [file ...]

DESCRIPTION
    The wc utility displays the number of lines, words, and bytes contained
    in each input file, or standard input (if no file is specified) to the
    standard output. A line is defined as a string of characters delimited
    by a {newline} character. Characters beyond the final {newline}
    character will not be included in the line count.

    A word is defined as a string of characters delimited by white space
    characters. White space characters are the set of characters for which
    the iswspace(3) function returns true. If more than one input file is
    specified, a line of cumulative counts for all the files is displayed on
    a separate line after the output for the last file.

    The following options are available:

    -c      The number of bytes in each input file is written to the standard
            output. This will cancel out any prior usage of the -m option.

    -l      The number of lines in each input file is written to the standard
            output.
```

Homework 3: Exercise 1

4. Try `file text.txt`. What does it report?
5. Try `wc text.txt`. Find out from the manpage what the three numbers reported mean. Screenshot it.
6. What's the **wc option** that prints the number of words only? Try it.

Homework 3: Exercise 1

6. nano text.txt.

- Type *Control*-G in the text editor to see the help text.
- You can type *Control*-X or *Control*-G to go back to displaying the contents of text.txt
- Ubuntu:
 - Scroll down and see if you can find the command for counting number of words, lines and characters.
 - What is that command and how do you type it? What is reported?
 - Screenshot its output for text.txt.
- macOS: nano is pico unfortunately:
 - The counting command is not directly available
 - Use *Control*-C to see the line number at your cursor position.
 - Screenshot the relevant output for text.txt.
- In either case, compare your answer with that obtained in 5.

Homework 3: Exercise 2

- Let's use the Terminal to make a frequency list of the words in `text.txt`
- First, look at the manpage for command `tr`.
- Next, let's replace all the punctuation characters by spaces.
- 1. Observe the output of both commands below. See any difference?

Which command do we want?

- `cat text.txt | tr '[:punct:]' ' '`
- `cat text.txt | tr -d '[:punct:]'`

note: a *space* here

- **Note 1:** pipe (`|`) sends the output of the `cat` command as input to the next command `tr`.

Homework 3: Exercise 2

- **Note 2:** we can redirect ('>') the output of the above command into a file, e.g. `text2.txt`, as follows:

- `command > text2.txt`

2. Next, we can put each word on a separate line using:

- `tr ' ' '\n'`

note: a *space* here

- **Note 3:** `\n` stands for a newline character.

3. Combine the previous two steps (1 & 2) together.

- **Note 4:** you can use `text2.txt` (*if you saved the output*), or just chain the command onto the end, i.e. do:

- `commands-so-far | tr ' ' '\n'`

Homework 3: Exercise 2

- Next, look at the manpage for command `uniq`.
4. Let's make a table of the frequency counts for each word using:
- `sort | uniq -c`
- by running the above command on the output of Step 3 above.
 - **Note 5:** you can chain this command as mentioned earlier, or save the output of step 4 into another file, e.g. `text3.txt`. Example:
 - `command | sort | uniq -c`
5. Why do we sort first? Read the `uniq` manpage to find out.

Homework 3: Exercise 2

6. Let's put the results in sorted order of frequency (*descending*) by appending another command:
 - `sort -rn`
 - to our list of commands so far (step 4 above).
 - Be sure to chain it using the pipe ('|').
 - consult the `sort` manpage to find out what the options `-r` and `-n` above do.
 - Show your output.

Instructions

- Email to sandiway@arizona.edu
- By Sunday midnight (*will be graded on Monday*)
- SUBJECT: 408/508 Homework 3: *YOUR NAME*
- **ONE** PDF file please
 - screenshots should be inside,
 - not as a bunch separate attachments
- do not submit Word .docx or .doc files
- do not submit a link to your PDF file, submit the actual file