Week 2 Lecture 1: More C

Review and Preview

- Basic C program structure
- First C programs
- Shell, directories, pipes

- Reviewing if
- Looping
- wcount.c program

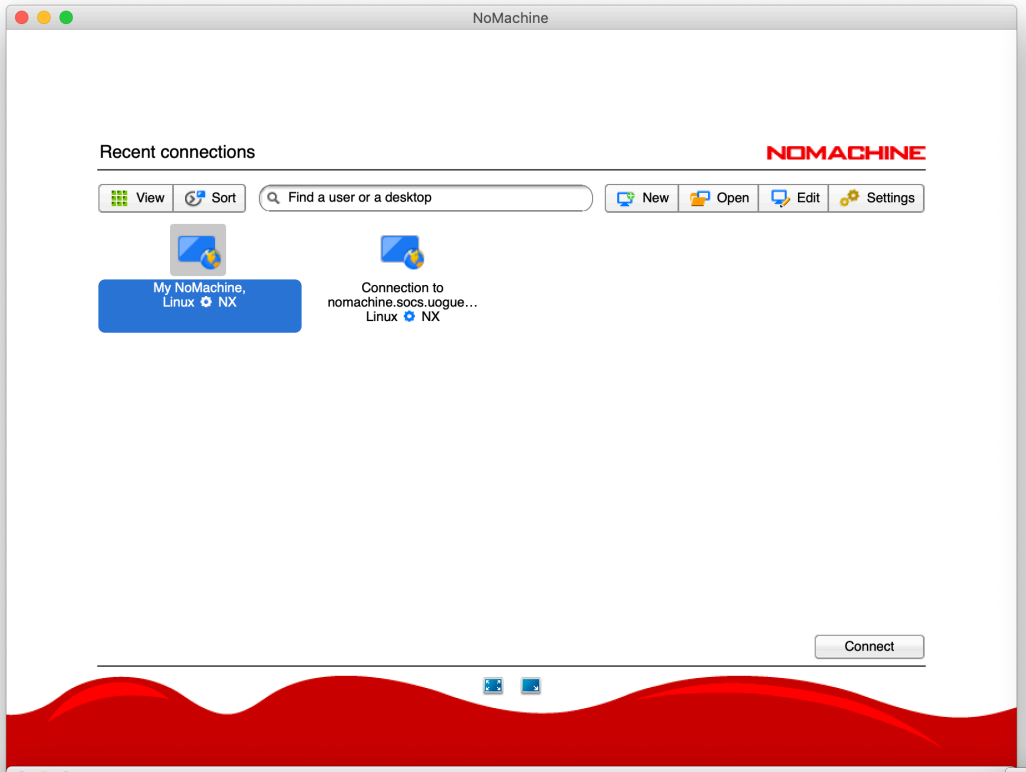
K&R: Chapter 1
Sections 1.1 to 1.6
Sections 2.1 to 2.6

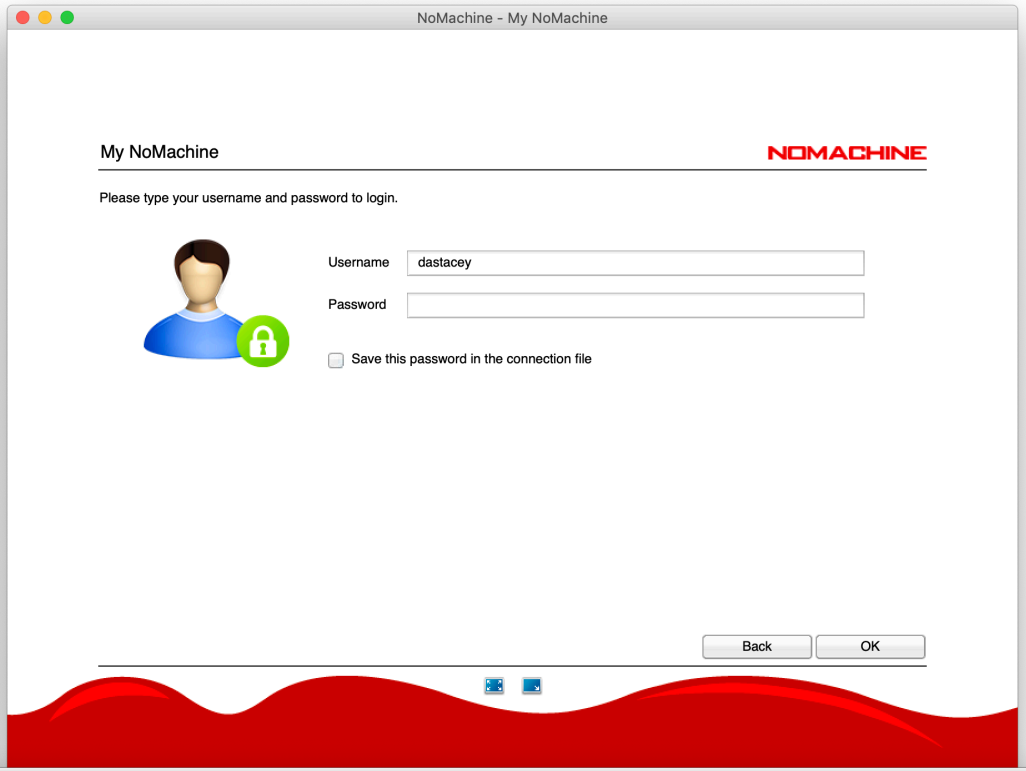
If you do not have a laptop…

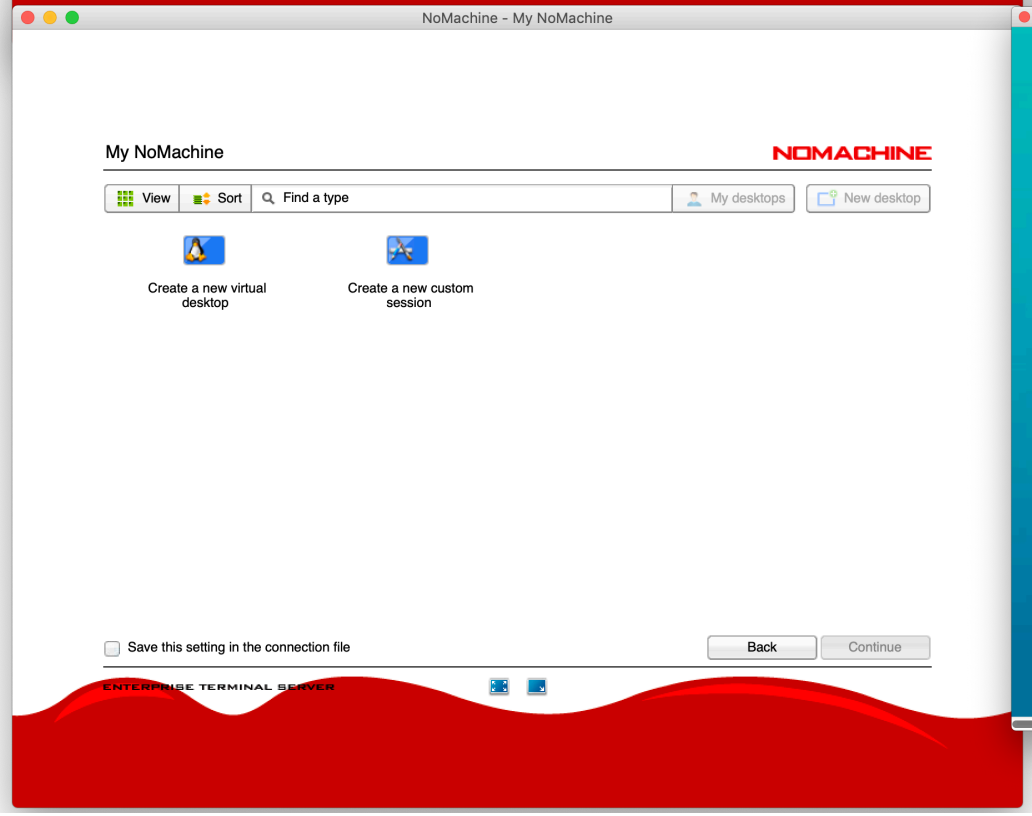
- You can use the SOCS No Machine server.
- Instructions for setting it up are on

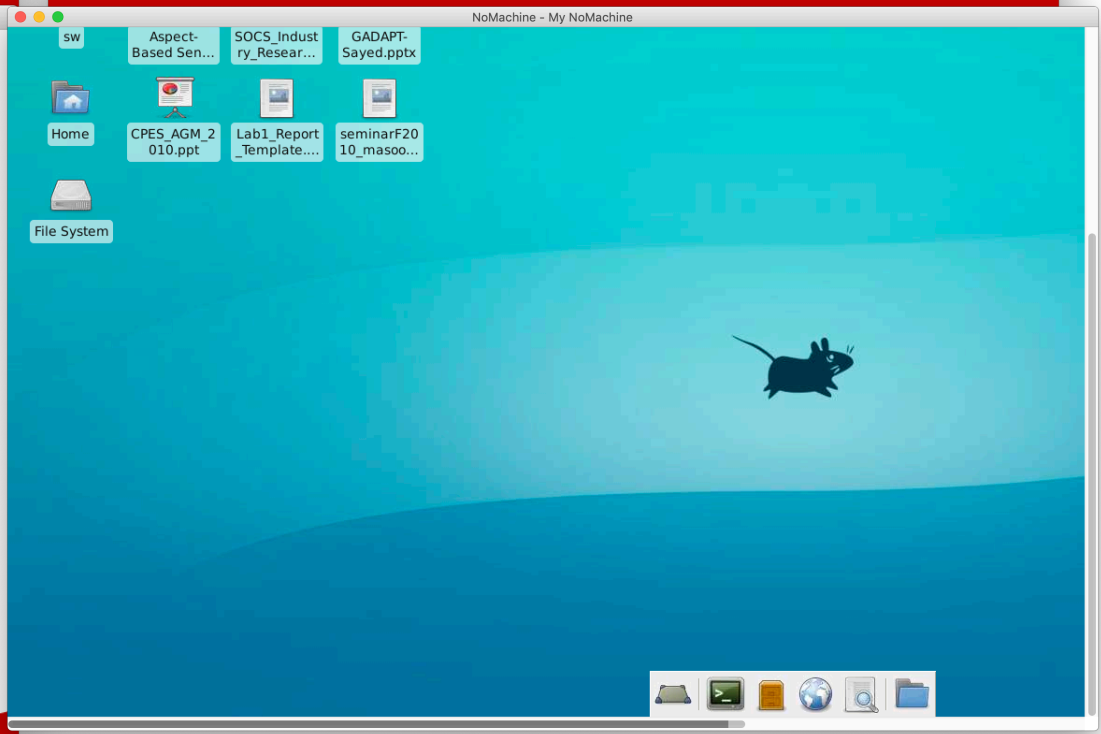
  https://wiki.socs.uoguelph.ca/techsupport/guides/nomachine
- A site that you should bookmark:

  https://wiki.socs.uoguelph.ca/


NoMachine
Recent connections
NOMACHINE
View
Sort
Find a user or a desktop
New
Open
Edit
Settings
My NoMachine,
Linux NX
Connection to
nomachine.socs.uogue…
Linux NX
Connect


NoMachine - My NoMachine
My NoMachine
NOMACHINE
Please type your username and password to login.
Username
dastacey
Password
Save this password in the connection file
Back
OK


NoMachine - My NoMachine
My NoMachine
NOMACHINE
View
Sort
Find a type
My desktops
New desktop
Create a new virtual
desktop
Create a new custom
session
Save this setting in the connection file
Back
Continue
ENTERPRISE TERMINAL SERVER


NoMachine - My NoMachine
sw
Aspect-
Based Sen...
SOCS_Indust
ry_Resear...
GADAPT-
Sayed.pptx
Home
CPES_AGM_2
010.ppt
Lab1_Report
_Template....
seminarF20
10_masoo...
File System

UNIX Command of the Day: `cal`

- The cal utility displays a simple calendar.
- If arguments are not specified, the current month is displayed.
- **`cal -m <number>`** displays the month (1..12) of the current year
- **`cal -m <number> <year>`** displays the month in the designated year.
- **`cal <year>`** displays the entire year

```
debs@deb-socs:~/CIS1300/FirstPrograms$ cal
   September 2019
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30

debs@deb-socs:~/CIS1300/FirstPrograms$ cal -m 10
    October 2019
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

debs@deb-socs:~/CIS1300/FirstPrograms$ cal -m 10 2020
    October 2020
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

debs@deb-socs:~/CIS1300/FirstPrograms$
```

```
debs@deb-socs:~/CIS1300/FirstPrograms$ cal 2019
                               2019
      January               February                March
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5                     1  2                  1  2
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   3  4  5  6  7  8  9
13 14 15 16 17 18 19  10 11 12 13 14 15 16  10 11 12 13 14 15 16
20 21 22 23 24 25 26  17 18 19 20 21 22 23  17 18 19 20 21 22 23
27 28 29 30 31        24 25 26 27 28        24 25 26 27 28 29 30
                                            31

       April                  May                   June
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6            1  2  3  4                     1
 7  8  9 10 11 12 13   5  6  7  8  9 10 11   2  3  4  5  6  7  8
14 15 16 17 18 19 20  12 13 14 15 16 17 18   9 10 11 12 13 14 15
21 22 23 24 25 26 27  19 20 21 22 23 24 25  16 17 18 19 20 21 22
28 29 30              26 27 28 29 30 31     23 24 25 26 27 28 29
                                            30

        July                  August               September
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6                  1  2  3   1  2  3  4  5  6  7
 7  8  9 10 11 12 13   4  5  6  7  8  9 10   8  9 10 11 12 13 14
14 15 16 17 18 19 20  11 12 13 14 15 16 17  15 16 17 18 19 20 21
21 22 23 24 25 26 27  18 19 20 21 22 23 24  22 23 24 25 26 27 28
28 29 30 31           25 26 27 28 29 30 31  29 30

      October               November              December
Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa  Su Mo Tu We Th Fr Sa
       1  2  3  4  5                     1  2   1  2  3  4  5  6  7
 6  7  8  9 10 11 12   3  4  5  6  7  8  9   8  9 10 11 12 13 14
13 14 15 16 17 18 19  10 11 12 13 14 15 16  15 16 17 18 19 20 21
20 21 22 23 24 25 26  17 18 19 20 21 22 23  22 23 24 25 26 27 28
27 28 29 30 31        24 25 26 27 28 29 30  29 30 31

debs@deb-socs:~/CIS1300/FirstPrograms$
```

Let's Review the `if` statement…

- The `if` statement allows you to tell the code to do different things depending on a condition(s).
- This is called **branching** - after the `if` statement you have two different pathways for the code to follow…and it can only follow one path.
- The basic structure is

```
if ( expr ) {
   Statements to execute if expr is true
} else {
   Statements to execute if expr is false
}
```

`if…else` Chains

```
if ( expr ) {
    Statements to execute if expr is true
} else if ( expr ) {
    Statements to execute if expr is true
} else {
    Statements to execute if none are true
}
```

`if…else` Operators

- **OR** - true when either boolean expression is **true**

```
if ( a > 10 || b > 10 ) …
```

- **AND** - true when all boolean expressions are **true**

```
if ( a > 10 && b > 10 ) …
```

- **NOT** - true when boolean expression is false

```
if ( !( a > 10 && b > 10 ) ) …
```

`if` Examples

```
if ( argc < 4 ) {
  printf (“Not enough cmdline args\n”);
}

if ( mm < 1 || mm > 12 ) {
  printf (“Month range is 1 to 12\n”);
  return (1);
}
```

`if` Examples

```
if ( mm == 1 && dd > 0 && dd < 32 ) {
  printf (“Day in January\n”);
}

if (mm == 9 || mm == 4 || mm == 6 || mm
== 11) {
  maxDD = 30;
}
```

`if` Question 1

```
int num1 = 5;
int num2 = 4;
if ( !(num1 > 10 && num2 > 10) ) {
  printf ( “Branch 1\n” );
} else {
  printf ( “Branch 2\n” );
}
???
```

`if` Question 2

```
int num1 = 11;
int num2 = 4;
if ( !(num1 > 10 && num2 > 10) ) {
  printf ( “Branch 1\n” );
} else {
  printf ( “Branch 2\n” );
}
???
```

`if` Question 3

```
int num1 = 11;
int num2 = 44;
if ( !(num1 > 10 && num2 > 10) ) {
  printf ( “Branch 1\n” );
} else {
  printf ( “Branch 2\n” );
}
???
```

Looping - repeat, repeat, …

The `while` Loop

Loops: Repeating Actions

- A basic tenet of programming is the repetition of actions - looping.
- A while loop executes while some condition is true.

```
while ( boolean expression ) {
    Statements to execute if the boolean
    expression is true
 }
```

https://www.programiz.com/c-programming/c-do-while-loops

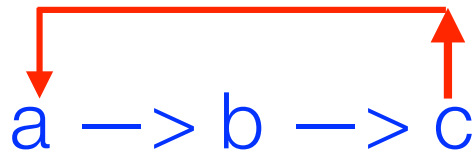
Loops: `while`

```
   a = 0;
a) while ( a < 10 ) {
b)     printf ( "%d \n", a );
b)     a = a + 2;
c) }
```

Test or Condition

Possible error

Output:
0
2
4
6
8



Loops: `while` Best Practices

- What if you wanted the previous example to print the numbers from 0 to 10?

```
while ( a <= 10 ) {
```

OR

```
while ( a < 11 ) {
```

- It is better to always use an **inequality** - consistency will help prevent errors in future code particularly when you make updates/changes.

The `for` loop - like `while` but more compact!

- The for loop allows us to specify the 3 important aspects of looping in one place:
  - **Start condition**: usually a counter, *e.g.* `i = 0`
  - **End condition**: when this test is false the looping stops, *e.g.* `i < 10`
  - How the start condition is changing while looping is happening, *e.g.* `i++`

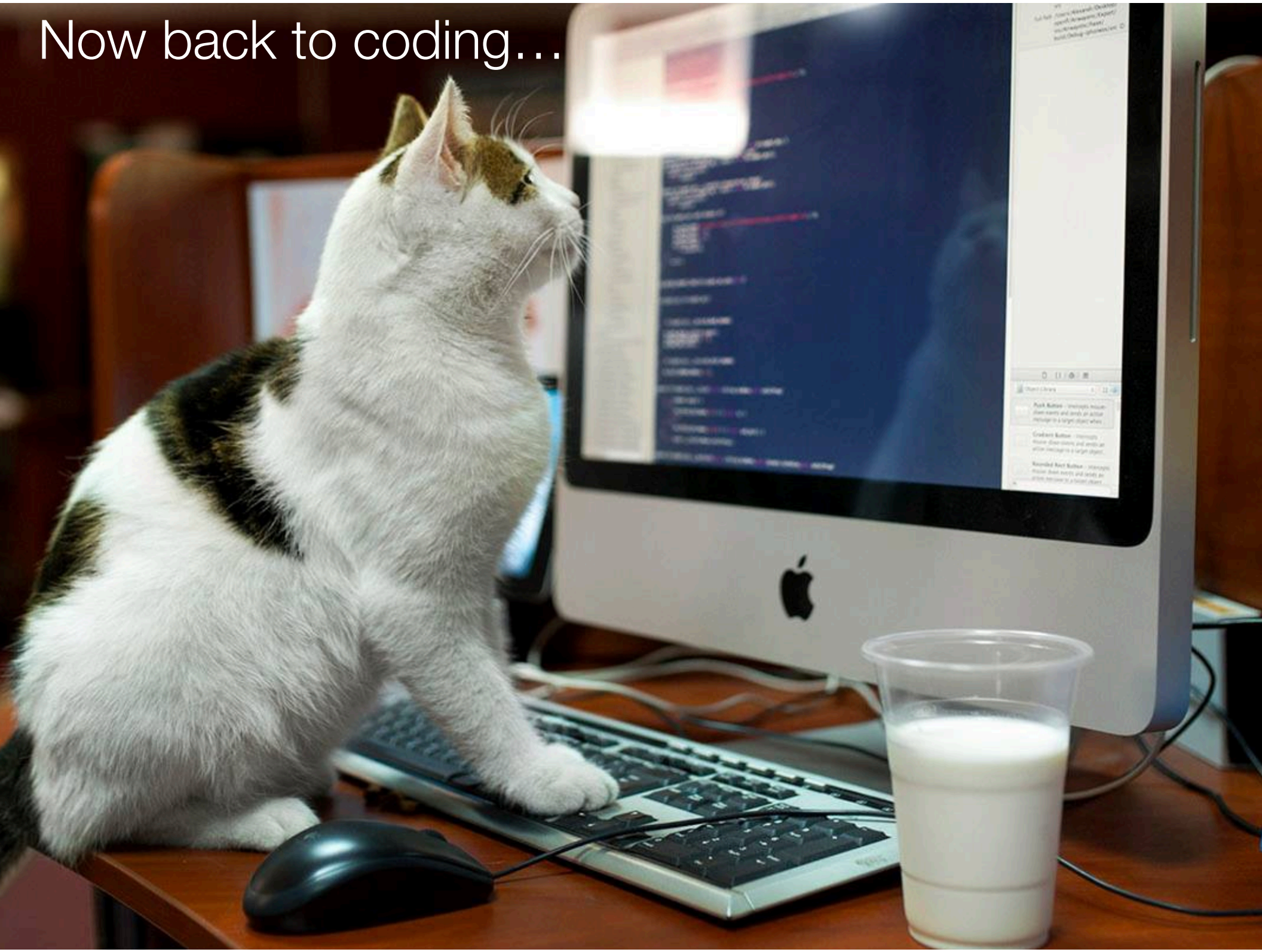
The `for` loop

```
for ( i=0; i<5; i++ ) {
   printf ( “i = %d\n", i );
}

$ ./counter
i = 0
i = 1
i = 2
i = 3
i = 4
```

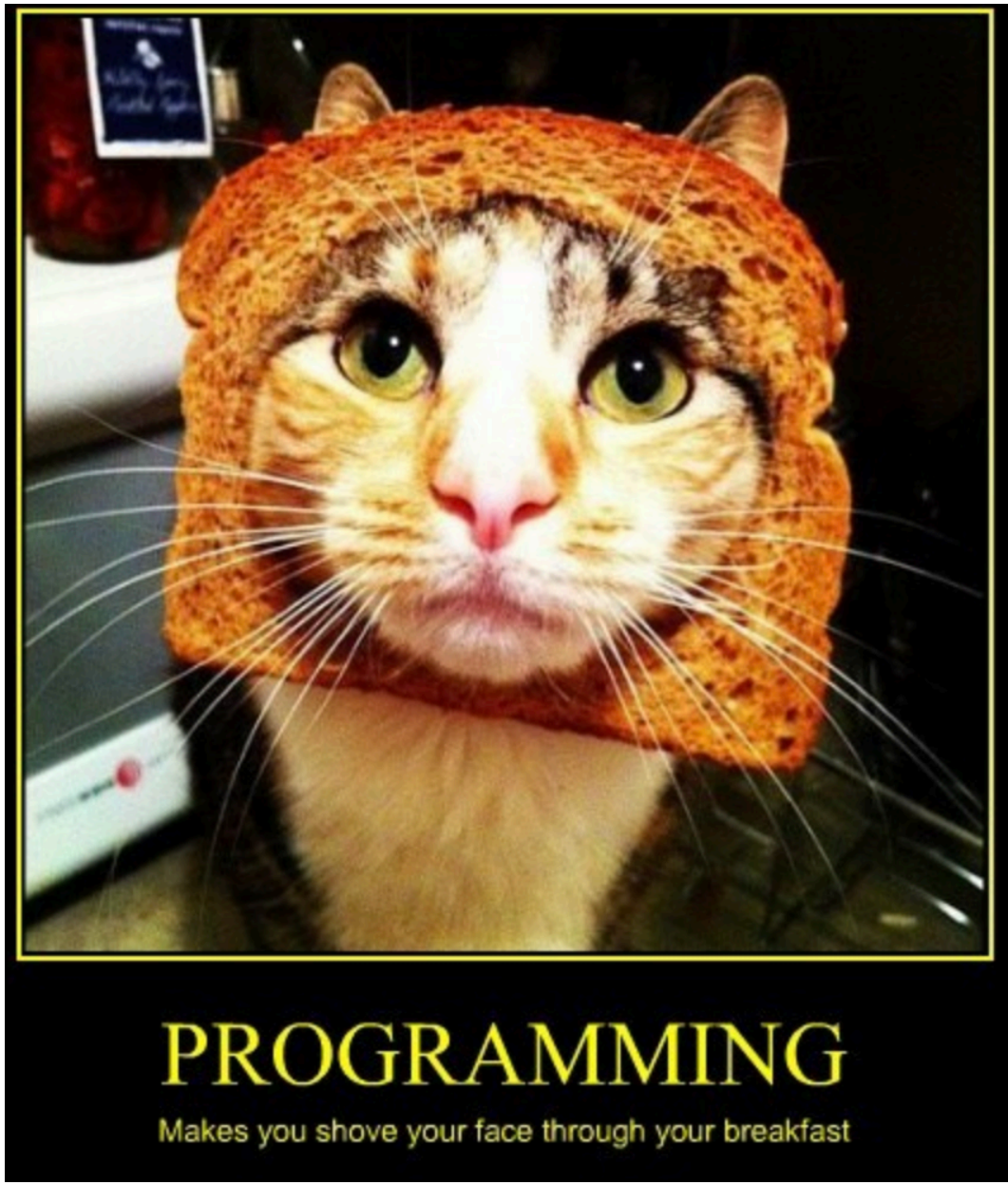
https://beginnersbook.com/2014/01/c-for-loop/

Now back to coding…



First C Programs

C Program #2: **Word Count** *(if…else, while,* ***STDIN****, Unix pipes)*



Program #2: Counting Characters

Function: To read in a text file from STDIN and count the number of characters, words, and lines and report on this.

K&R - Section 1.5.4 (page 20)

Program Design / Algorithm

- Read in the input, one character at a time **until** there is no more input and each time a character is read in, *increment* the **character counter**.
- `if` the character is a *Newline* character
  - *increment* the **line counter**.
- `if` the character is a Space, Newline, or Tab
  - then you are now **outside** of a word,
- `else if` the character is **NOT** a Space, Newline, **or** Tab **and** you were outside of a word
  - then you are now inside of a word and you increment the **word counter**.

a)

b)

c)

```
#include <stdio.h>

#define IN  1   /* inside a word */
#define OUT 0   /* outside a word */

/*
 * Program Name: wcount.c
 * Author(s): Deb Stacey
 * Date of Last Update: August 4, 2019
 * Purpose: reads input from stdin and counts the lines, words, and
 *          sentences and prints these counts to stdout.
 *          Modern version of code on page 20 of K&R (section 1.5.4).
 *          Similar to Linux command wc.
 * Testing: testfile(s): testWC1.txt
 *          protocol: compare to wc results
 * Change Log:
 */
```

a) Preprocessor directive: standard I/O include

b) Preprocessor directive: define statements

c) Comments

Declare a variable that stores 1 character

```
int main ( int argc, char *argv[] ) {

   char c;

   int numLines = 0;   /* number of lines */
   int numWords = 0;   /* number of words */
   int numChars = 0;   /* number of characters */

   int state = OUT;

   while ( (c = getchar()) != EOF ) {
      numChars++;
         .
         .
         .
         .
   }
   printf ( "%d %d %d\n", numLines, numWords, numChars );

   return ( 0 );
}
```

Declare and initialize variables for counts

Declare a variable that stores whether we are currently inside a word

Loop: read a char from stdin until there are no more

Different Ways to do the Same Thing

```
while ( (c = getchar()) != EOF ) {
    …
}
```

```
c = getchar();
while ( c != EOF ) {
    …
    c = getchar();
}
```

Inside the Loop

Remember that the variable **`state`** *starts as* **`OUT`**

```
numChars++;

if ( c == '\n' ) {
    numLines++;
}

if ( c == ' ' || c == '\n' || c == '\t' ) {
    state = OUT;
} else if (state == OUT) {
    state = IN;
    numWords++;
}
```

`if` the character is a *Newline* character

`if` the character is a Space, Newline, or Tab

`else if` the character is **NOT** a Space, Newline, **or** Tab **and** you were outside of a word

You were outside of a word but now you have found a letter, i.e. another word, so increment the word count and set **`state`** to be inside of a word.

```
int main ( int argc, char *argv[] ) {
   char c;
   int numLines = 0;  /* number of lines */
   int numWords = 0;  /* number of words */
   int numChars = 0;  /* number of characters */

   int state = OUT;

   while ( (c = getchar()) != EOF ) {
      numChars++;

      if ( c == '\n' ) {
         numLines++;
      }

      if ( c == ' ' || c == '\n' || c == '\t' ) {
         state = OUT;
      } else if (state == OUT) {
         state = IN;
         numWords++;
      }
   }
   printf ( "%d %d %d\n", numLines, numWords, numChars );

   return ( 0 );
}
```

Newline: \n

Tab: \t

Update char count

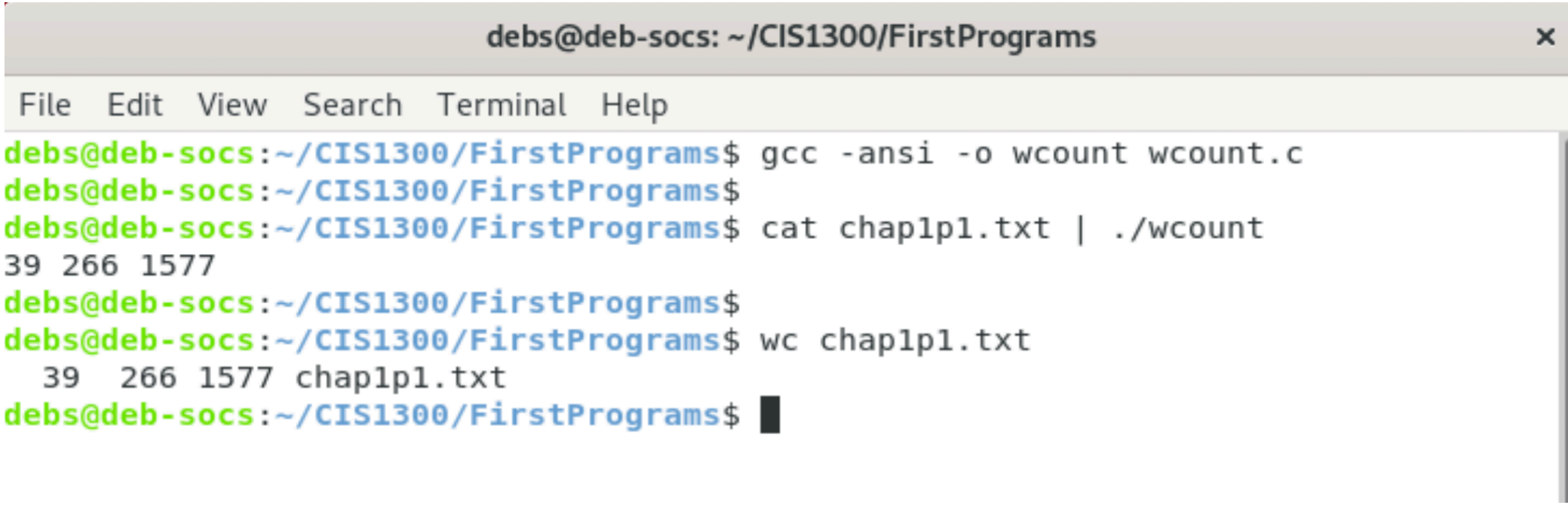
Update line count

Update word count

	T	**h**	**i**	**s**		**i**	**s**		**o**	**n**	**e**	**.**	**\n**
OUT	IN	IN	IN	IN	OUT	IN	IN	OUT	IN	IN	IN	IN	OUT
Char	1	2	3	4	5	6	7	8	9	10	11	12	13
Word	1	1	1	1	1	2	2	2	3	3	3	3	3
Line	0	0	0	0	0	0	0	0	0	0	0	0	1

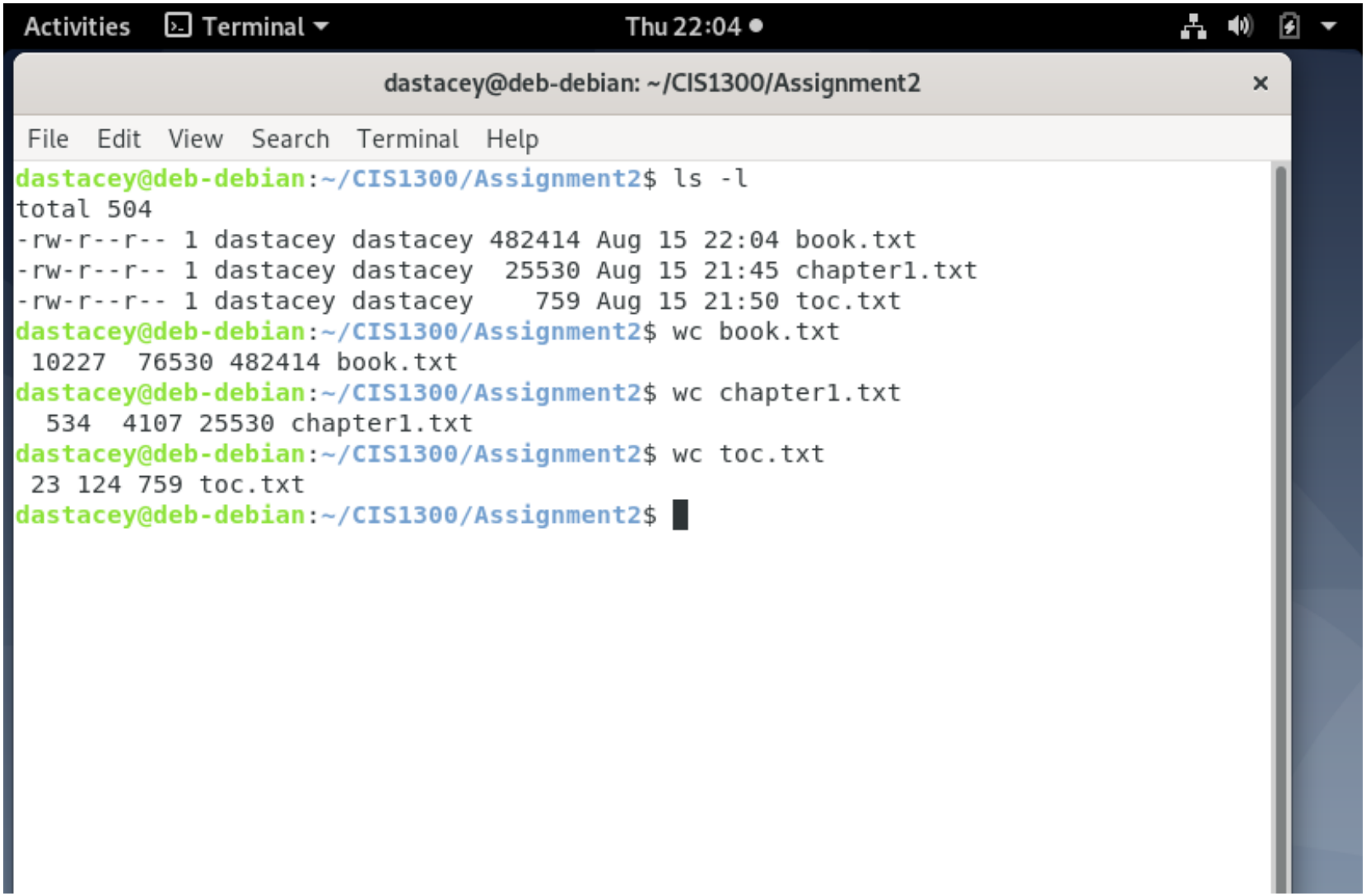
	L	**a**	**s**	**t**		**l**	**i**	**n**	**e**	**.**	**\n**
	IN	IN	IN	IN	OUT	IN	IN	IN	IN	IN	OUT
Char	14	15	16	17	18	19	20	21	22	23	24
Word	4	4	4	4	4	5	5	5	5	5	5
Line	1	1	1	1	1	1	1	1	1	1	2

Testing against the shell command wc



`wc` - character, word and line counts

- print newline, word, and byte counts for each file



Review and Hotwash

What we covered today and what you thought was important.

https://www.amazon.com/Review-My-Cat-Tanner-Ringerud/dp/1402285361