CSE 1320

Week of 03/20/2023

Instructor : Donna French

```
#include <stdio.h>
#include <string.h>

#define MAX 80

int main(void)
{
  char *Token
  char Buffer[

  printf("Ente
  fgets(Buffer

  // strtok()

  return 0;
}
```

```
Token = strtok(Buffer, ",");
printf("The first word is %s\n", Token);
Token = strtok(NULL, ",");
printf("The second word is %s\n", Token);
Token = strtok(NULL, ",");
printf("The third word is %s\n", Token);
```

More Tools for Our Toolbox

`islower(ch)`	tests if `ch` is a lowercase alphabetic character
`isupper(ch)`	tests if `ch` is an uppercase alphabetic character
`isalpha(ch)`	tests if `ch` is an alphabetic character
`isalnum(ch)`	tests if `ch` is an alphanumeric character
`isdigit(ch)`	tests if `ch` is a decimal digit
`ispunct(ch)`	tests if `ch` is punctuation character
`isspace(ch)`	tests if `ch` is a whitespace character

```
#include <ctype.h>
```

```c
int main(void)
{
  char ch;

  printf("Enter a character ");
  scanf("%c", &ch);

  if (islower(ch)) printf("islower\n");
  if (isupper(ch)) printf("isupper\n");
  if (isalpha(ch)) printf("isalpha\n");
  if (isalnum(ch)) printf("isalnum\n");
  if (isdigit(ch)) printf("isdigit\n");
  if (ispunct(ch)) printf("ispunct\n");
  if (isspace(ch)) printf("isspace\n");

  return 0;
}
```

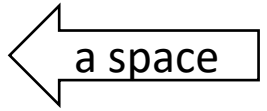
```
Enter a character a
islower
isalpha
isalnum

Enter a character A
isupper
isalpha
isalnum

Enter a character 1
isalnum
isdigit

Enter a character !
ispunct

Enter a character
isspace
```



ischDemo.c

More Tools for Our Toolbox

`tolower(ch)` returns the lowercase version of `ch`

`toupper(ch)` returns the uppercase version of `ch`

```
char ch;
char chUP;
char chLOW;

printf("Enter a character ");
scanf("%c", &ch);

chUP = toupper(ch);
printf("ch %c has been changed to %c\n", ch, chUP);

printf("ch %c has been changed to %c\n", ch, tolower(ch));
```

```
#include <ctype.h>
```

```
Breakpoint 1, main () at touplowDemo.c:10
10              printf("Enter a character ");
(gdb) step
11              scanf("%c", &ch);
(gdb)
Enter a character a
13              chUP = toupper(ch);
(gdb) p ch
$4 = 97 'a'
(gdb) step
14              printf("ch %c has been changed to %c\n", ch, chUP);
(gdb) p ch
$5 = 97 'a'
(gdb) p chUP
$6 = 65 'A'
(gdb) step
ch a has been changed to A
16              printf("ch %c has been changed to %c\n", ch, tolower(ch));
(gdb) step
ch a has been changed to a
18              return 0;
```

must include `<stdlib.h>`

More Tools for Our Toolbox

Two new library functions

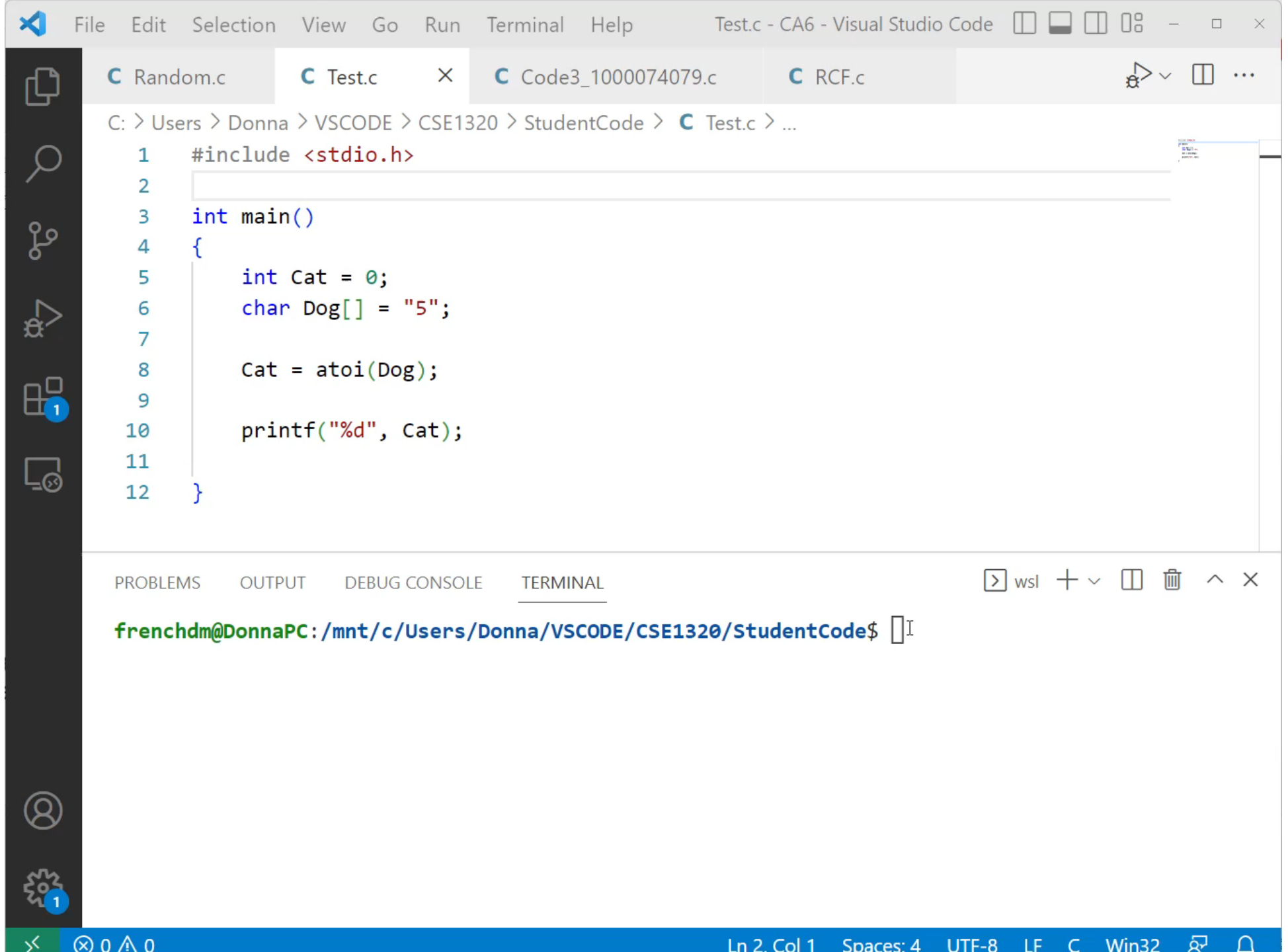
`atoi()` and `atof()`

`atof()` takes a null terminated string containing the ASCII representation of a floating point number as its parameter and converts the string to the corresponding value of type `float` and returns that value.

`atoi()` takes a null terminated string containing the ASCII representation of an integer number as its parameter and converts the string to the corresponding value of type `int` and returns that value.

```
15                  printf("Enter a float value ");
(gdb)
16                  fgets(Input, 100, stdin);
(gdb)
Enter a float value 21.9
18                  MyFloatVar1 = atof(Input);
(gdb) p Input
$1 = "21.9\n", '\000' <repeats 94 times>
(gdb) step
23                  printf("MyFloatVar1 value is %f\n", MyFloatVar1);
(gdb) p MyFloatVar1
$2 = 21.8999996
(gdb) step
MyFloatVar1 value is 21.900000
```

```
25                printf("\n\nEnter an integer value ");
(gdb)
26                fgets(Input, 100, stdin);
(gdb)
Enter an integer value 12
28                MyIntVar1 = atoi(Input);
(gdb) p Input
$3 = "12\n\000\n", '\000' <repeats 94 times>
(gdb) step
30                printf("MyIntVar1 value is %d\n", MyIntVar1);
(gdb) p MyIntVar1
$4 = 12
(gdb) step
MyIntVar1 value is 12
```


File Edit Selection View Go Run Terminal Help
Test.c - CA6 - Visual Studio Code
Random.c
Test.c
Code3_1000074079.c
RCF.c
C: > Users > Donna > VSCODE > CSE1320 > StudentCode > Test.c > ...
#include <stdio.h>
int main()
{
int Cat = 0;
char Dog[] = "5";
Cat = atoi(Dog);
printf("%d", Cat);
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
wsl
frenchdm@DonnaPC:/mnt/c/Users/Donna/VSCODE/CSE1320/StudentCode$
Ln 2, Col 1
Spaces: 4
UTF-8
LF
C
Win32

More Tools for Our Toolbox

`memcmp()` and `memcpy()` go byte by byte regardless of what is in those bytes.

`strcpy()` and `strcmp()` look for null terminators

Two new library functions

`memcpy()` and `memcmp()`

`memcpy()` is a lot like `strcpy()` except that it does not rely on a null terminator – it is given the number of bytes to copy

`memcmp()` is a lot like `strcmp()` except that it does not rely on a null terminator – it is given the number of bytes to compare

memcpy() and memcmp()

```
char Array1[100] = {"The quick fox jumps"};
char Array2[100];
char Array3[100];

memcpy(Array2, Array1, strlen(Array1));
memcpy(Array3, Array1, strlen(Array1)+1);

printf("Array1  %s\nArray2  %s\nArray3
        Array1, Array2, Array3);
```

```
Array1  The quick fox jumps
Array2  The quick fox jumps▒*
Array3  The quick fox jumps
```

Array2 does not include \0

```
(gdb) p Array2
$1 = "The quick fox jumps\252\252*\000\000\020\350\377\377\377\177\000\000\000ч\
252*\000\000<\003@\000\000\000\000\000v", '\000' <repeats 39 times>"\377, \265\3
60\000\000\000\000\000\302\000\000"
```

Array3 does include \0

```
 (gdb) p Array3
$2 = "The quick fox jumps\000\000\000\000\000\002\223\000\311>", '\000' <repeats
 11 times>"\340, \366\252\252\252*\000\000\001", '\000' <repeats 15 times>, "\00
1\000\000\000\000\000\000\000`\310Q\311>\000\000\000\000\347\377\377\377\177\000
\000\000\000\000\000\000\000\000\000v\000\000"
```

memcpy() and memcmp()

```
char Array4[100] = {"hello\0\0\0"};
char Array5[100] = {"hello\0!!"};

if (memcmp(Array4, Array5, 8) == 0)
     printf("equal\n");
else
     printf("not equal\n");

if (strcmp(Array4, Array5) == 0)
     printf("equal");
else
     printf("not equal\n");
```

memcmp() returns 0 if the two arrays are equal for the number of characters compared

\0 is not equal to !

only compares up to the \0

memcpy()

```
char Array1[100] = {"The quick fox jumps"};
char Array2[100] = {};

memcpy(&Array1[4], "brown", 5);
```

The brown fox jumps

```
memcpy(&Array1[strlen("The quick ")], "dog park", 3);
```

The brown dog jumps

```
memcpy(&Array1[14], "shakes", strlen("jumps")+1);

strcat(Array2, Array1);
```

Did we get lucky here?

```
printf("%s", Array2);
```

The brown dog shakes

memset

```
void *memset(void *str, int c, size_t n)
```

Parameters

str	This is a pointer to the block of memory to fill.
c	This is the value to be set. The value is passed as an `int`, but the function fills the block of memory using the `unsigned char` conversion of this value.
n	This is the number of bytes to be set to the value.

memset

```
17          char MyTestArray[] = {"ABCDE"};
(gdb) p MyTestArray
$1 = "ABCDE"

21          memset(MyTestArray, ' ', sizeof(MyTestArray));

(gdb) p MyTestArray
$2 = "     "
```

memset

```
17          char MyTestArray[] = {"ABCDE"};
(gdb) p MyTestArray
$1 = "ABCDE"

21          memset(MyTestArray, '\0', sizeof(MyTestArray));

(gdb) p MyTestArray
$2 = "\000\000\000\000\000"
```

memset

```
17          char MyTestArray[] = {"ABCDE"};
(gdb) p MyTestArray
$1 = "ABCDE"

21          memset(MyTestArray, 'A', sizeof(MyTestArray));

(gdb) p MyTestArray
$2 = "AAAAA"
```

memset

```
char MyTestArray[] = {"ABCDE"};
char A1[10] = {"XYZ"};
char A2[10] = {"XYZ"};

strcat(A1, MyTestArray);
printf("%s\n", A1);

memset(MyTestArray, 'A', sizeof(MyTestArray));

strcat(A2, MyTestArray);
printf("%s", A2);
```

sizeof(MyTestArray) is 6 because of \0

XYZABCDE
XYZAAAAAAA

?

should have used strlen()

What is the sizeof(MyTestArray)?

memset

```
17          int MyTestArray[] = {1,2,3,4,5};
(gdb) p MyTestArray
$1 = {1, 2, 3, 4, 5}

21          memset(MyTestArray, 0, sizeof(MyTestArray));

(gdb) p MyTestArray
$2 = {0, 0, 0, 0, 0}
```

memset

```
17          int MyTestArray[] = {1,2,3,4,5};
(gdb) p MyTestArray
$1 = {1, 2, 3, 4, 5}

21          memset(MyTestArray, 1, sizeof(MyTestArray));

(gdb) p MyTestArray
$2 = {16843009, 16843009, 16843009, 16843009, 16843009}
```

memset

```
17          int MyTestArray[] = {1,2,3,4,5};
(gdb) p/x MyTestArray
$1 = {0x1, 0x2, 0x3, 0x4, 0x5}

21          memset(MyTestArray, 1, sizeof(MyTestArray));

(gdb) p/x MyTestArray
$2 = {0x1010101, 0x1010101, 0x1010101, 0x1010101, 0x1010101}
```

function fills the block of memory using the `unsigned char` conversion of the value so it put `01` in each BYTE of the `int` so `4 01`'s

memset

$17_{10} = 11_{16}$

$-1_{10} = FF_{16}$

```
memset(MyTestArray, 17, sizeof(MyTestArray));

(gdb) p MyTestArray
$2 = {286331153, 286331153, 286331153, 286331153, 286331153}

(gdb) p/x MyTestArray
$3 = {0x11111111, 0x11111111, 0x11111111, 0x11111111, 0x11111111}

memset(MyTestArray, -1, sizeof(MyTestArray));

(gdb) p MyTestArray
$1 = {-1, -1, -1, -1, -1}
(gdb) p/x MyTestArray
$2 = {0xffffffff, 0xffffffff, 0xffffffff, 0xffffffff, 0xffffffff}
```

`memset`

Summary

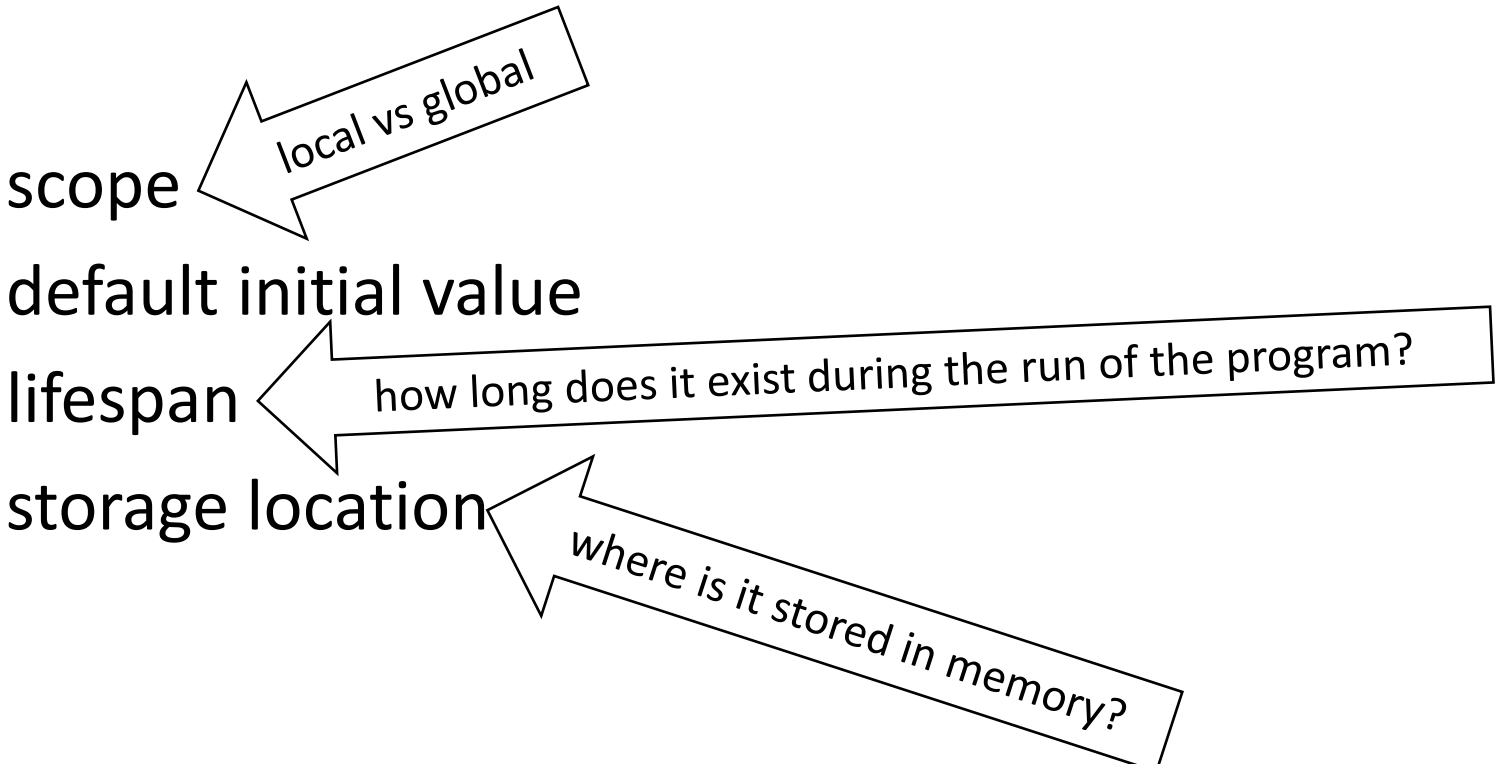
Only use 0 and -1 with `memset()` and integer arrays.

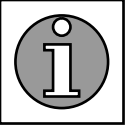
Be very careful using `memset()` with character arrays – be aware of the null terminator.

Automatic versus Static Variables

Storage Class

Storage classes are used to describe the features of a variable or function.





Automatic versus Static Variables

Automatic Variables

```
auto int IntVar;
   int IntVar;
```

- default storage class
- automatic variables are created each time its function is called and destroyed when the execution of its function terminates
- when an automatic variable is created without being initialized, it is not given an initial value – may contain garbage.
- when an automatic variable is created with an initialization, the initialization is done each time the variable is created

Automatic versus Static Variables

`auto`

scope

- inside function - local

default initial value

- contain garbage until explicitly initialized

lifespan

- created when function called and destroyed when function exits

storage location

- stack

Automatic versus Static Variables

Static Variables

```
static int IntVar;
```

- static variables exist the whole time the program is executing
- memory space is allocated when program starts and is deallocated when program ends
- static variables are given the default initial value of 0
- if an initializer is used, then the variable is initialized once at the beginning of the program

Automatic versus Static Variables

`static`

scope

inside function - local

default initial value

0

lifespan

created when program starts and ends when program ends

storage location

data segment

Automatic versus Static Variables

`auto`

scope
- inside function - local

default initial value
- contain garbage until explicitly initialized

lifespan
- created when function called and destroyed when function exits

storage location
- stack

`static`

scope
- inside function - local

default initial value
- 0

lifespan
- created when program starts and ends when program ends

storage location
- data segment

```
void CallMyFunction(void)
{
   static int staticVar1;
   static int staticVar2 = 100;

   int autoVar1;
   int autoVar2 = 100;

   printf("Value of staticVar1 = %d\n", staticVar1++);
   printf("Value of staticVar2 = %d\n", staticVar2++);
   printf("Value of autoVar1   = %d\n", autoVar1++);
   printf("Value of autoVar2   = %d\n", autoVar2++);
}
```

`staticVar1` is initialized to 0 for us and `staticVar2` is set to 100 once and both retain their values between function calls and are not reset.

`autoVar1` is system trash and `autoVar2` is set to 100 every time the function is called.

```
int i;

for (i = 0; i < 3; i++)
{
   CallMyFunction();
}
```

```
Value of staticVar1 = 0
Value of staticVar2 = 100
Value of autoVar1   = -920532032
Value of autoVar2   = 100

Value of staticVar1 = 1
Value of staticVar2 = 101
Value of autoVar1   = -920543041
Value of autoVar2   = 100
```

System trash

System trash

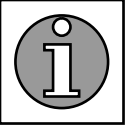
```
Value of staticVar1 = 2
Value of staticVar2 = 102
Value of autoVar1   = -920632138
Value of autoVar2   = 100
```

System trash incremented

```
Address of staticVar1 = 0x600a54
Address of staticVar2 = 0x600a44
Address of autoVar1   = 0x7fff26ebc28c
Address of autoVar2   = 0x7fff26ebc288
```

data segment

stack



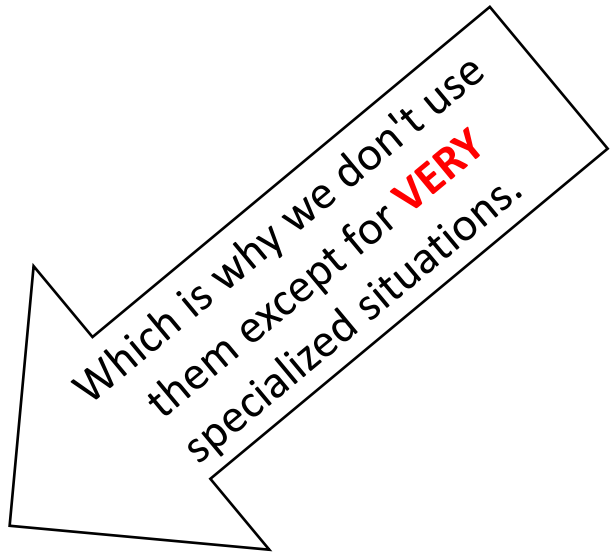
Automatic versus Static Variables

Register Variables

```
register int i;
```

- programmer requests that a variable be placed in a register
- usually indicates that a variable will be used frequently
  - improve speed and performance – indicies and loop counters
- no guarantee that the variable will be placed in the register
- very limited in availability and size
- illegal to use the address operator & with the name of a register variable

```
printf("%p", &i);
error: address of register variable ‘i’ requested
```



```
void print_it(void)
{
   register int i;
   int x;

   i = 12345;
   x = 98765;
   printf("i = %d", i);
}
```

calling `print_it()` from `main()`

```
print_it () at registerDemo.c:9
9                i = 12345;
(gdb)
10               x = 98765;
(gdb) p &i
Address requested for identifier
"i" which is in register $rsi
(gdb)
```

register

```
void print_it(void)
{
   int i;
   int x;

   i = 12345;
   x = 98765;
   printf("i = %d", i);
}
```

calling `print_it()` from `main()`

```
print_it () at registerDemo.c:10
10               i = 12345;
(gdb)
11               x = 98765;
(gdb) p &i
$3 = (int *) 0x7fffffffe788
```

stack memory

Global versus Local Variables

Local Variables

- only known inside the function block or compound statement block in which they were defined

- can be legally referenced at any point from its declaration to the closing braces for that block or function

Global versus Local Variables

Global Variables

- variable that can be referenced by more than one function
- defined outside function or compound statement blocks
- global variables are defined before all functions in a source code file
- global variables can be referenced by all functions in that file
- global variables are in existence during the full execution time of the program

```
int Pongo;
int Perdita;

void Dog(int Puppy)
{
  int Patch;
  int Lucky;
  Pongo = Perdita;
}

void Spots(int Puppy)
{
  int Rolly;
  int Penny;
  Pongo = Perdita;
}

int main(void)
{
  int Freckles;
  int Pepper;

  Dog(Freckles);
  Spots(Pepper);

  Pongo = Perdita;
  Freckles = Lucky;
  Pepper = Penny;

  return 0;
}
```

Global

```
lvsgDemo.c:30: error: 'Lucky' undeclared
lvsgDemo.c:31: error: 'Penny' undeclared
```

lvsgDemo.c

Global versus Local Variables

CAUTION

Global variables should be used with discretion.

All functions can access global variables and change their values.

The effect of a function changing a variable from outside its scope is called a

side effect

Every change to a global variable is a side effect.

sideeffectDemo.c

```
int X = 0;  /* Global version of X */

void SetXFunction(void)
{
  X = 987;
}

void PrintXFunction(void)
{
  printf("PrintXFunction()\tX = %d\n", X);
}

void NewSetXFunction(int *NewX)
{
  X = 567;
}
```

```
int main(void)
{
  /* Local version of X */
  int X = 123;

  printf("main()       X = %d\n", X);
  SetXFunction();
  printf("main()       X = %d\n", X);
  PrintXFunction();
  NewSetXFunction(&X);
  PrintXFunction();
  printf("main()       X = %d\n", X);

  return 0;
}
```

```
main()                    X = 123
main()                    X = 123
PrintXFunction()          X = 987
PrintXFunction()          X = 567
main()                    X = 123
```

CRLF vs LF vs CR

First time on a computer after using typewriter

memes
@memes4the2000s

Someday in the near future, people won't laugh at this because they don't get it.

CRLF vs LF vs CR

CRLF

Windows

LF

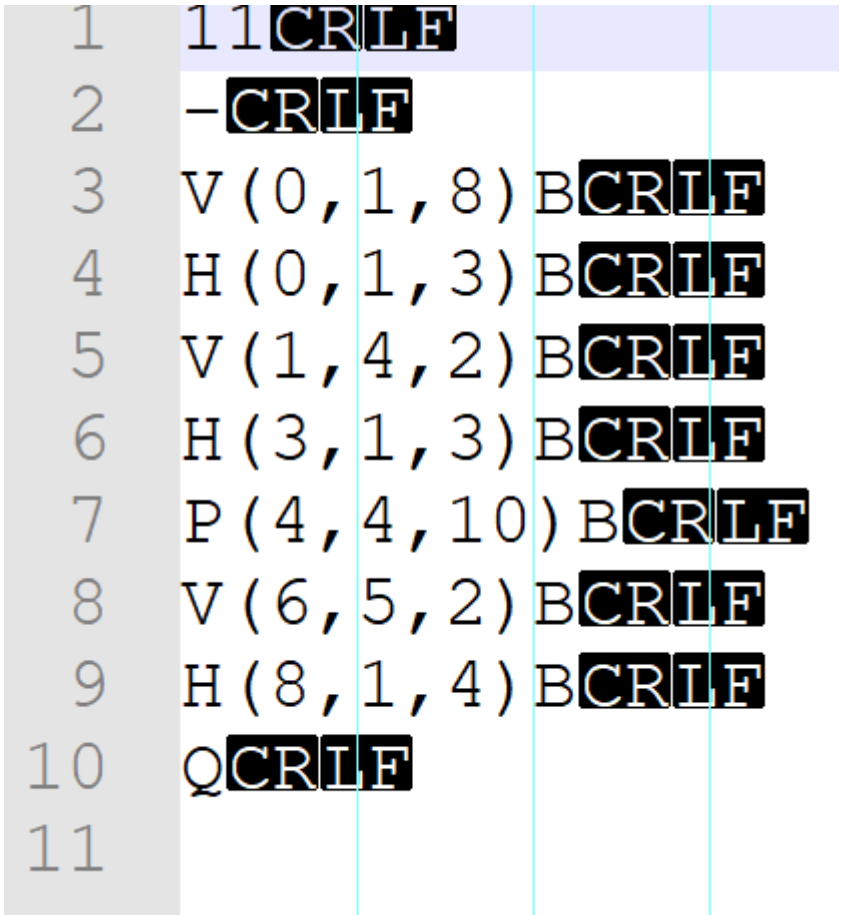
Unix

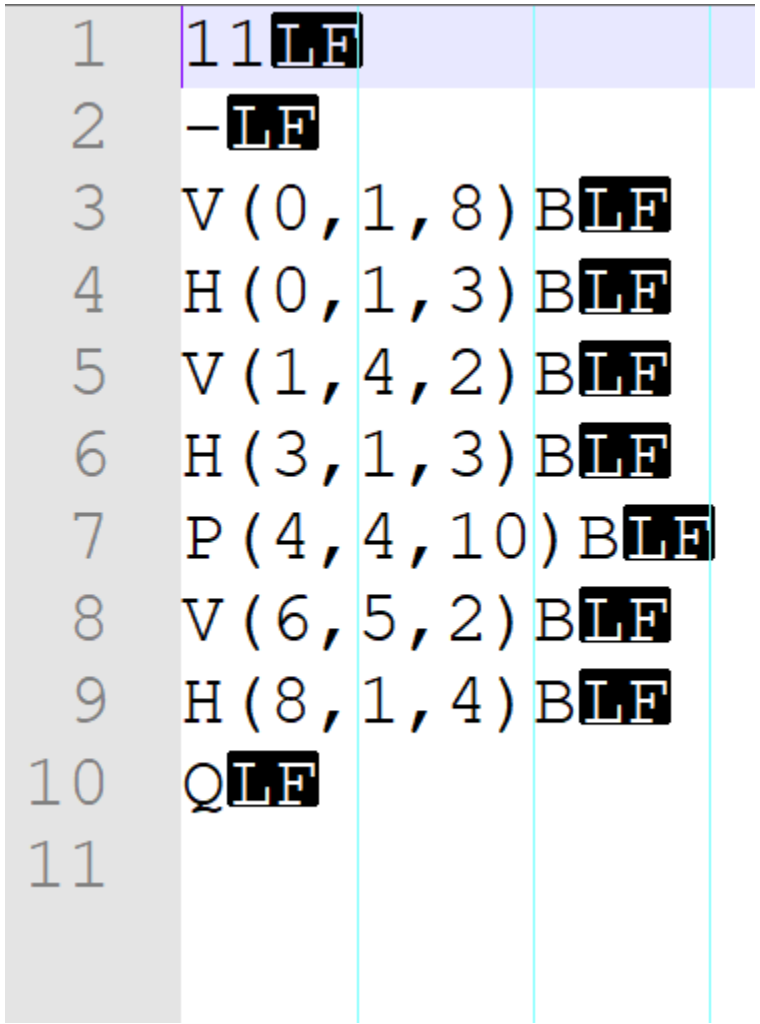
CR

Mac

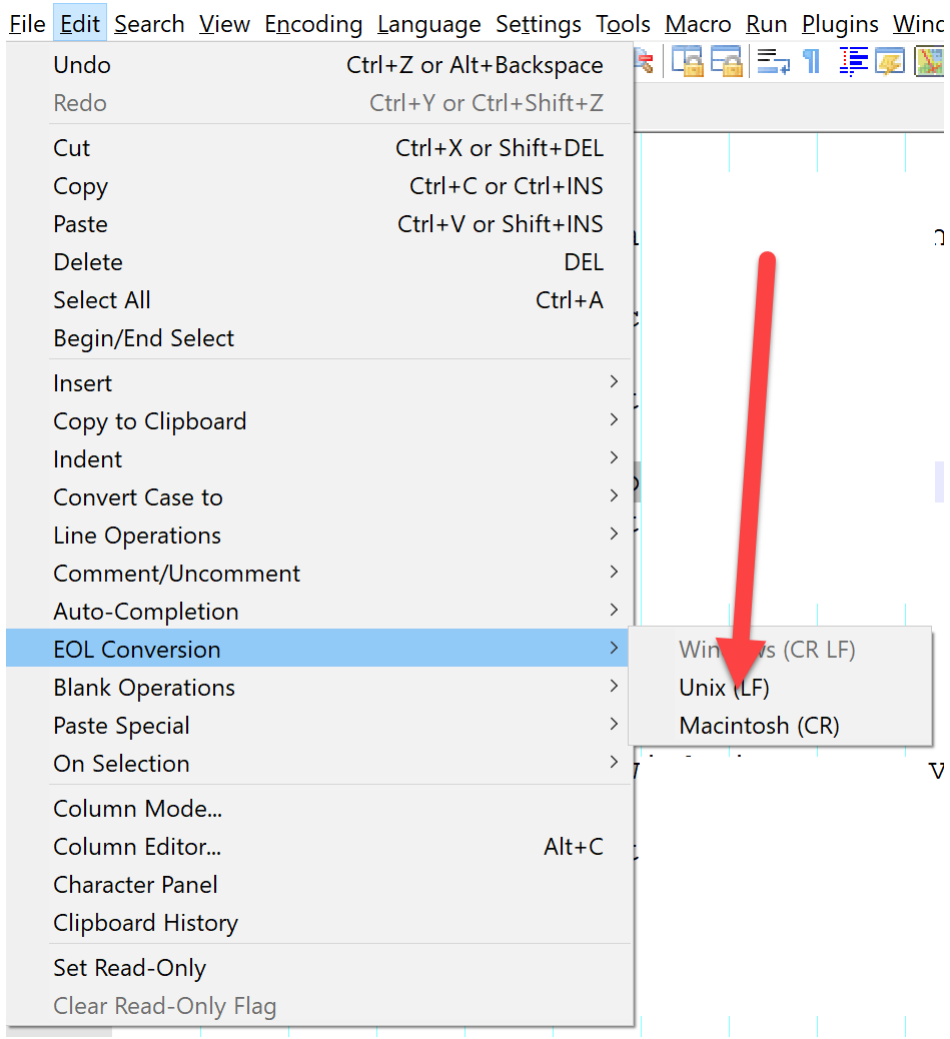
Ascii	Char	Ascii	Char	Ascii	Char	Ascii	Char
0	Null	32	Space	64	@	96	`
1	Start of heading	33	!	65	A	97	a
2	Start of text	34	"	66	B	98	b
3	End of text	35	#	67	C	99	c
4	End of transmit	36	$	68	D	100	d
5	Enquiry	37	%	69	E	101	e
6	Acknowledge	38	&	70	F	102	f
7	Audible bell	39	'	71	G	103	g
8	Backspace	40	(	72	H	104	h
9	Horizontal tab	41	)	73	I	105	i
10	Line feed	42	*	74	J	106	j
11	Vertical tab	43	+	75	K	107	k
12	Form feed	44	,	76	L	108	l
13	Carriage return	45	-	77	M	109	m
14	Shift in	46	.	78	N	110	n
15	Shift out	47	/	79	O	111	o
16	Data link escape	48	0	80	P	112	p
17	Device control 1	49	1	81	Q	113	q
18	Device control 2	50	2	82	R	114	r
19	Device control 3	51	3	83	S	115	s
20	Device control 4	52	4	84	T	116	t
21	Neg. acknowledge	53	5	85	U	117	u
22	Synchronous idle	54	6	86	V	118	v
23	End trans. block	55	7	87	W	119	w
24	Cancel	56	8	88	X	120	x
25	End of medium	57	9	89	Y	121	y
26	Substitution	58	:	90	Z	122	z
27	Escape	59	;	91	[	123	{
28	File separator	60	<	92	\	124	|
29	Group separator	61	=	93	]	125	}
30	Record separator	62	>	94	^	126	~
31	Unit separator	63	?	95	_	127	Forward del.

UNIX Line Feeds





UNIX Line Feeds




```
sed -i.old 's/\r$//' input.txt
```

CRLF vs LF vs CR

```
cat file.txt | tr '\r' '\n' | tr -s '\n' > file.translated.txt
```

This Unix command will translate the
CR in Mac files or the CRLF in Windows files to
UNIX LF

You can also use

```
sed -i.old 's/\r$//' MyFile.txt
```

Omega Server

Omega (Ω) is available for UTA student academic use. It is a general purpose UNIX server suitable for learning software development. Users typically login to the server using SSH or SFTP tools. Most operating systems have SSH client built in.

```
$ ssh netid@omega.uta.edu
```

Frequently Asked Questions

1. How do I get an account on Omega?
2. What is my URL on omega.uta.edu?
3. How much space can I have?
4. Where can I learn about omega.uta.edu and Unix in general?
5. How do I use CGI scripts on omega.uta.edu?
6. Where can I go for more help with HTML and webpage building?
7. What operating system is omega.uta.edu using?
8. What software is available on omega.uta.edu?

Question: How do I get an account on omega.uta.edu? Every student and employee has an Omega account by default and the account does not need to be requested.

Question: What operating system is omega.uta.edu using?

Red Hat Enterprise Linux 5.11 release 5.11 (Tikanga) running on four Intel Xeon CPU E5-2699 v4 @ 2.20GHz processors with 16GB of RAM.

Question: What software is available on omega.uta.edu?

Available tools include gcc/g++/gfortran 4.1.2, Python 2.4.3, PHP 5.1.6, ruby 1.8.5, MySQL 14.14, SQL*Plus 11.2.0.4, SML 110.74, Mathematica 7.0/8.0, perl 5.8.8, Java 1.6.0r20, cmake 2.6p4.

https://uta.service-now.com/kb_view.do?sysparm_article=KB0010329

VIRTUAL PRIVATE NETWORK

SERVICE AUDIENCE

Students | Staff | Faculty | Research

Experience the journey through the virtual private network that offers a safe and encrypted connection over the internet. It ensures sensitive university and personal data is transmitted securely between your device and UTA systems. * #1 Recommended Service from OIT

* **PULSE SECURE: With over 15 years of innovation and refinement**, Pulse Secure is built for the next generation of faculty, staff, and students as **it offers the following cutting-edge features**: extended sessions, performance stability, and seamless connection to UTA resources. Any faculty, staff, or students can enjoy this premier service from Pulse Secure.

For more information, please go to the OIT Knowledge Base or use the links located on this article.

KNOWLEDGE BASE ARTICLES

→ Pulse Secure Resource Guide

Omega

PC and Mac

- Download FileZilla
- Tool for moving (FTPing) files between your computer and Omega

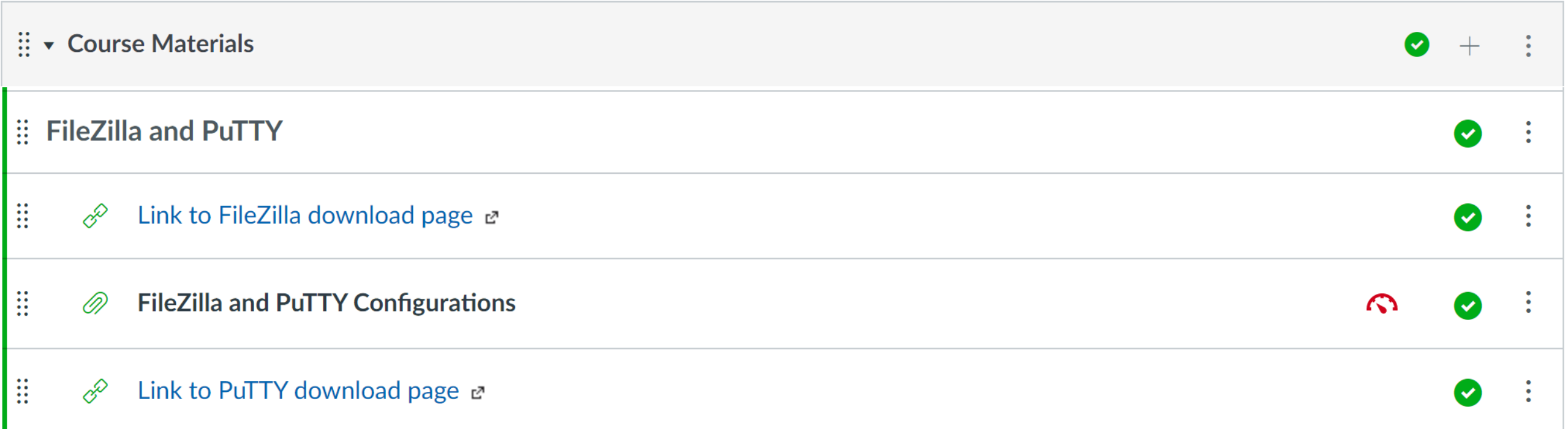
PC

- Download PuTTY (optional)

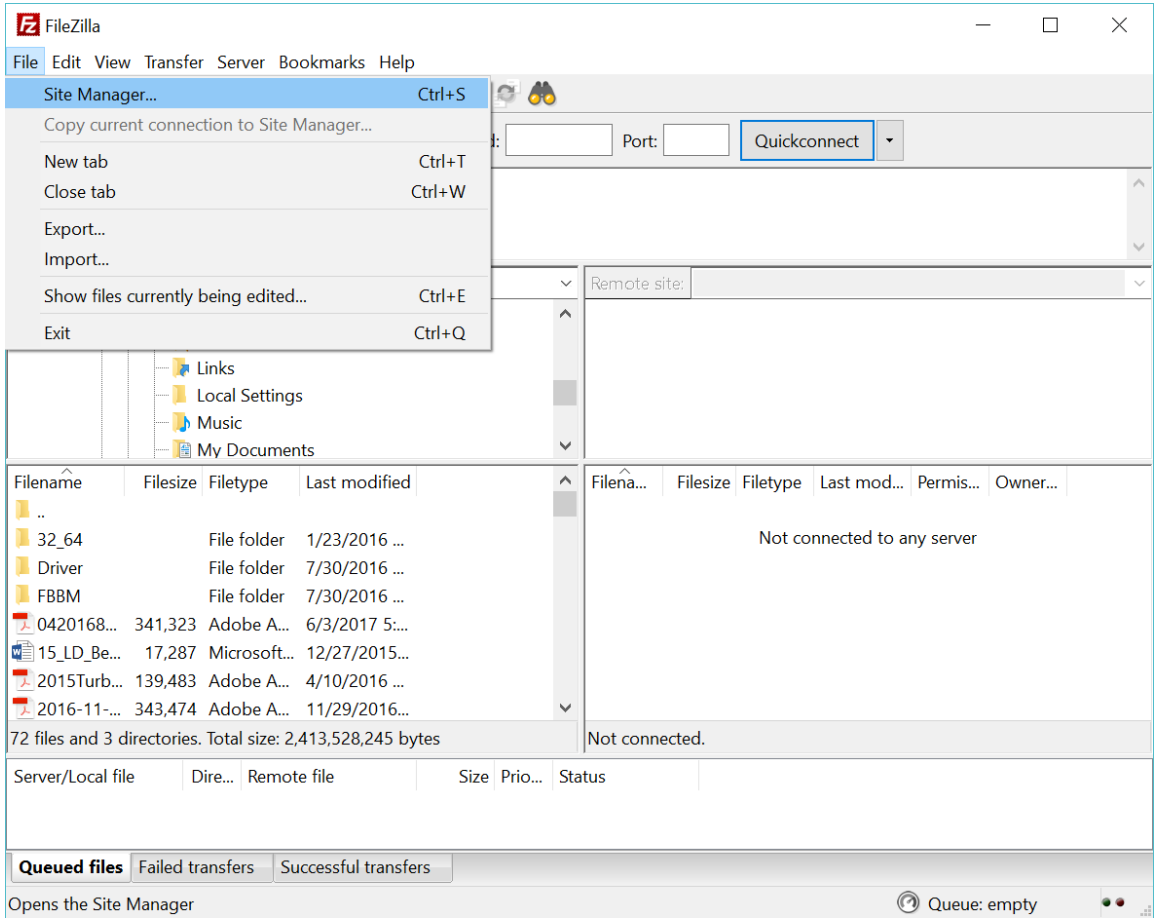
PC and Mac

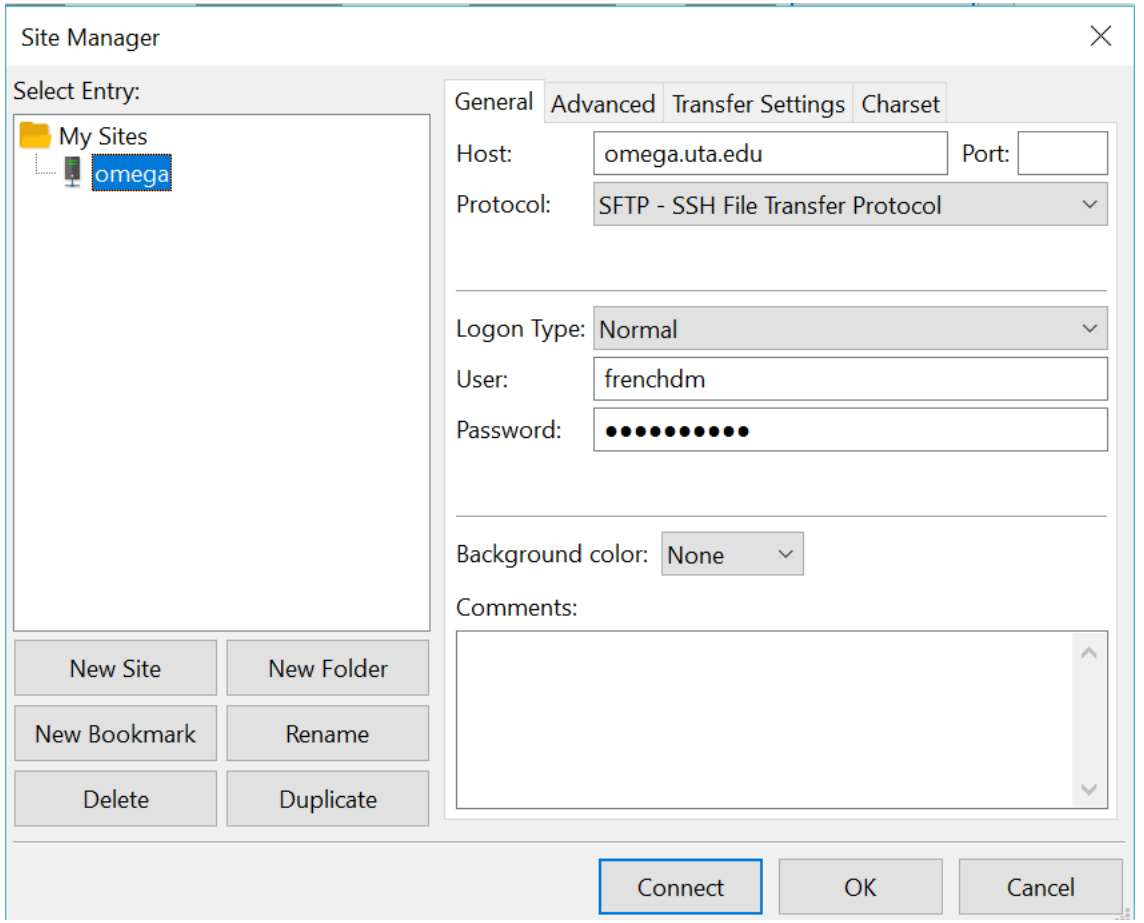
- Use SSH to connect to Omega

Omega

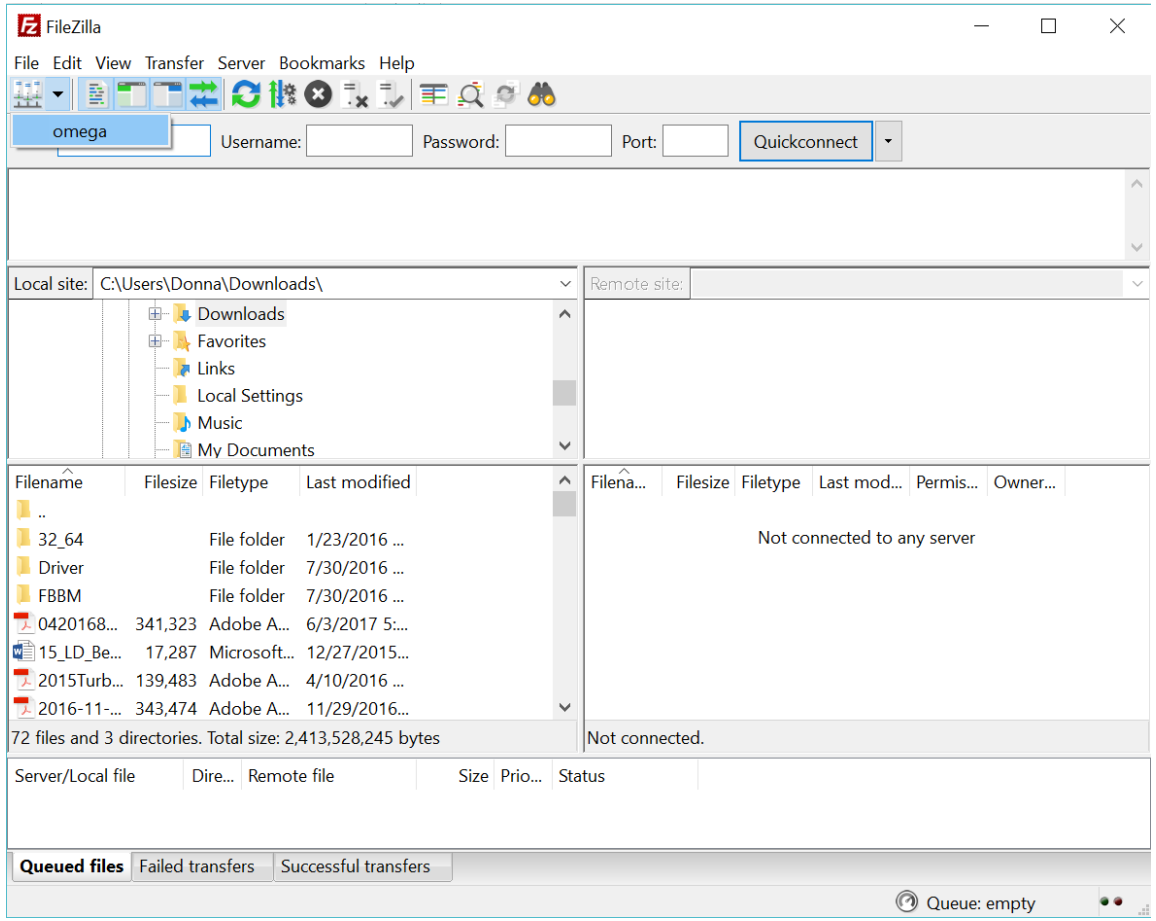

Course Materials
FileZilla and PuTTY
Link to FileZilla download page
FileZilla and PuTTY Configurations
Link to PuTTY download page

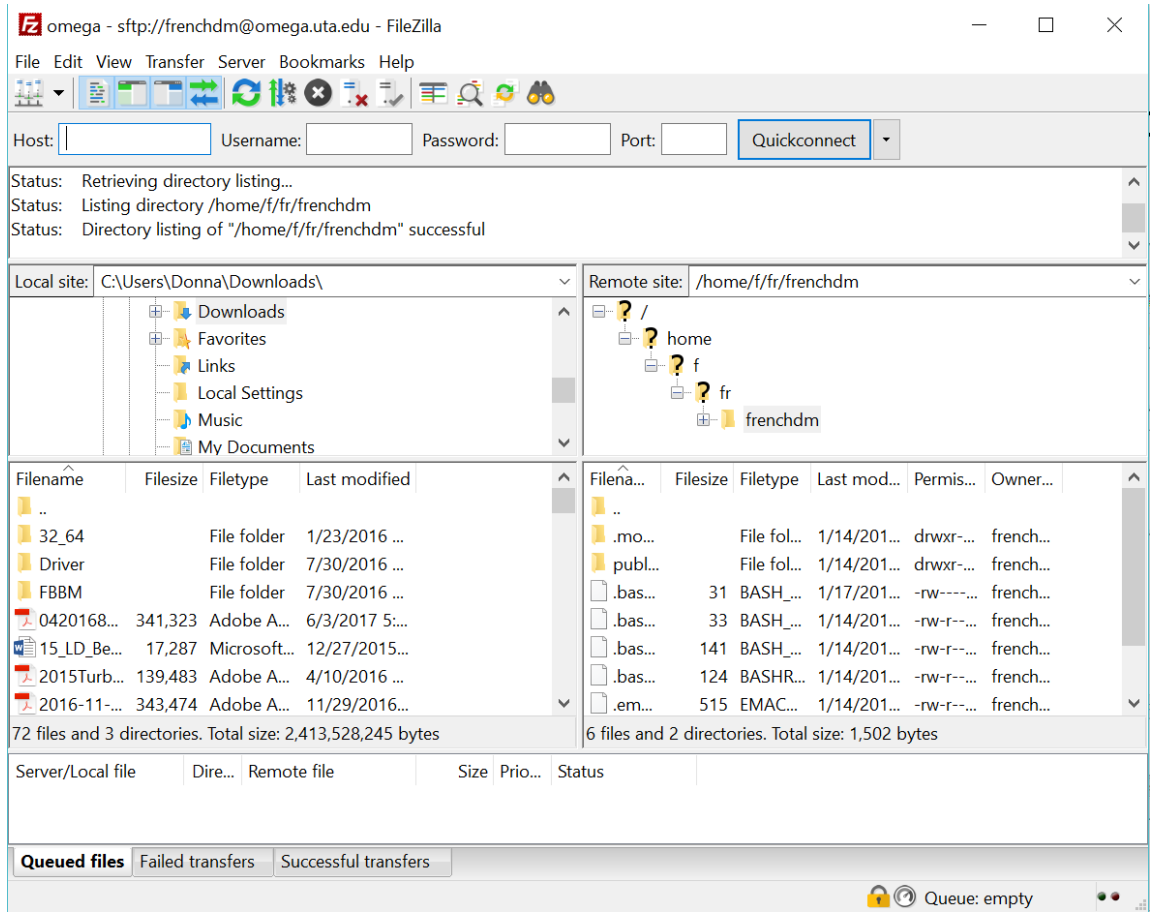
FileZilla
FTP Tool for both PC and Mac



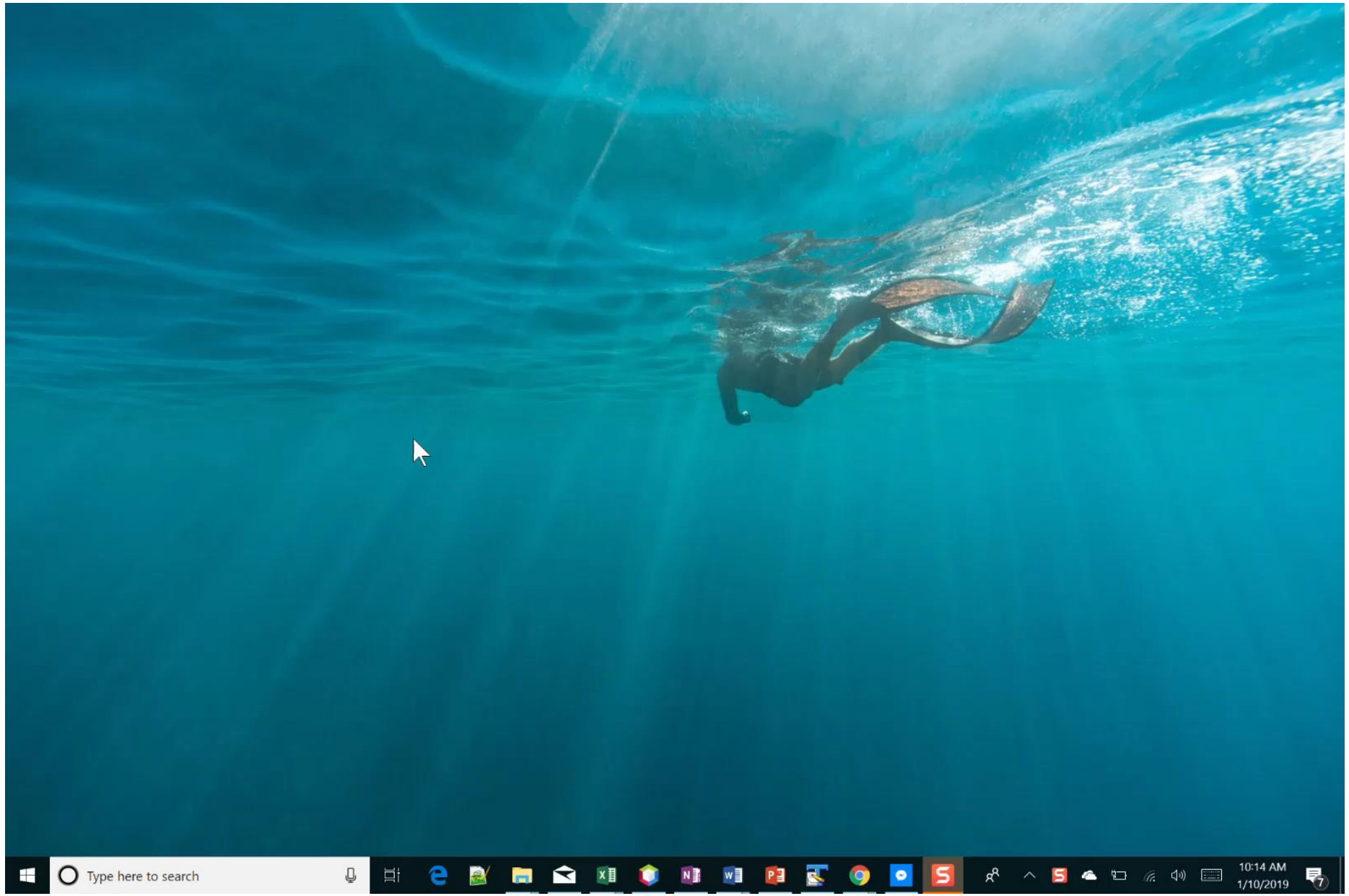


FileZilla

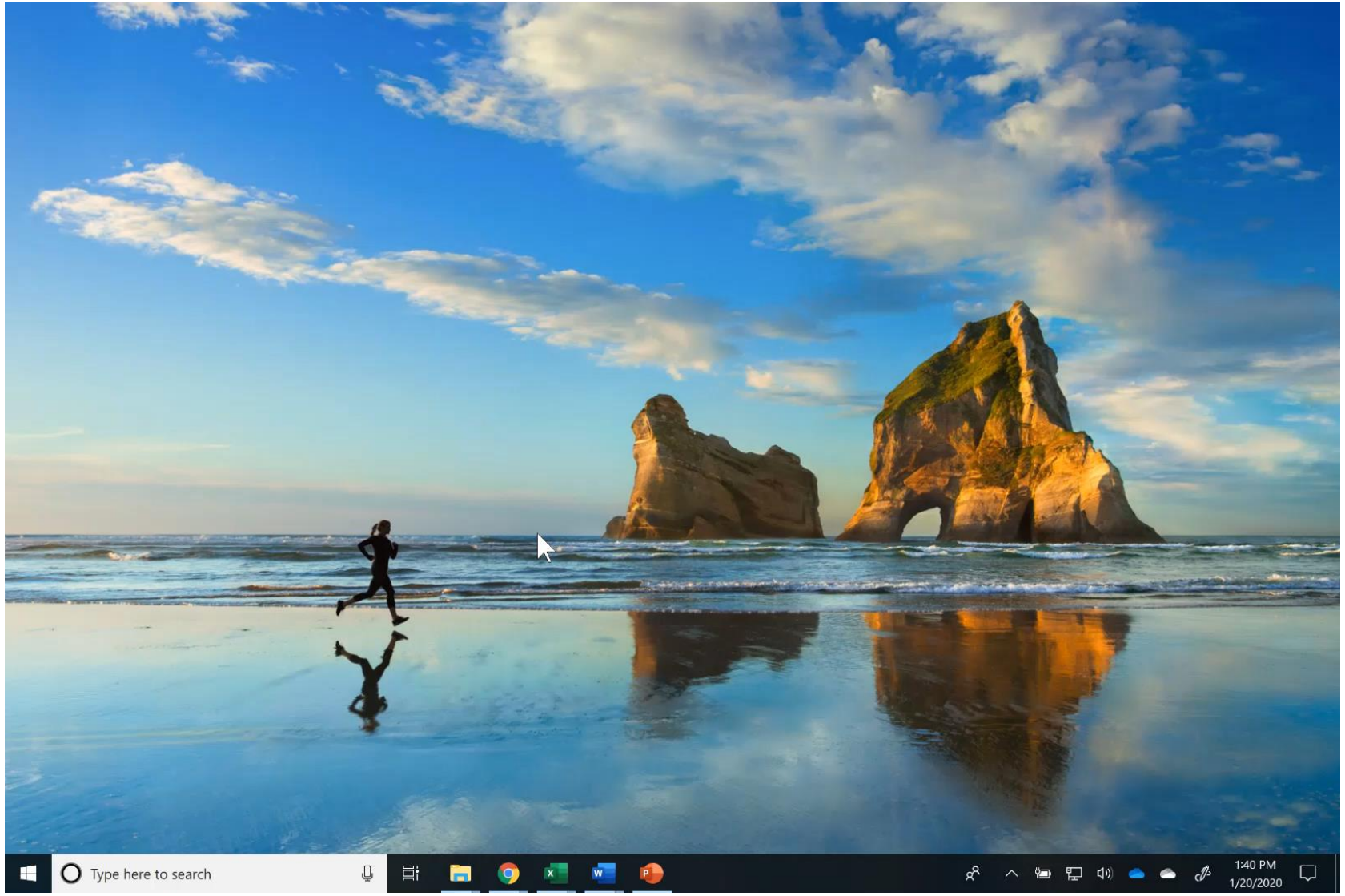

FileZilla
File Edit View Transfer Server Bookmarks Help
omega
Username:
Password:
Port:
Quickconnect
Local site: C:\Users\Donna\Downloads\
Remote site:
Downloads
Favorites
Links
Local Settings
Music
My Documents
Filename Filesize Filetype Last modified
32_64 File folder 1/23/2016 ...
Driver File folder 7/30/2016 ...
FBBM File folder 7/30/2016 ...
0420168... 341,323 Adobe A... 6/3/2017 5:...
15_LD_Be... 17,287 Microsoft... 12/27/2015...
2015Turb... 139,483 Adobe A... 4/10/2016 ...
2016-11-... 343,474 Adobe A... 11/29/2016...
Not connected to any server
72 files and 3 directories. Total size: 2,413,528,245 bytes
Not connected.
Server/Local file Dire... Remote file Size Prio... Status
Queued files Failed transfers Successful transfers
Queue: empty


omega - sftp://frenchdm@omega.uta.edu - FileZilla
File Edit View Transfer Server Bookmarks Help
Host:
Username:
Password:
Port:
Quickconnect
Status: Retrieving directory listing...
Status: Listing directory /home/f/fr/frenchdm
Status: Directory listing of "/home/f/fr/frenchdm" successful
Local site: C:\Users\Donna\Downloads\
Remote site: /home/f/fr/frenchdm
Downloads
Favorites
Links
Local Settings
Music
My Documents
/
home
f
fr
frenchdm
Filename Filesize Filetype Last modified
32_64 File folder 1/23/2016 ...
Driver File folder 7/30/2016 ...
FBBM File folder 7/30/2016 ...
0420168... 341,323 Adobe A... 6/3/2017 5:...
15_LD_Be... 17,287 Microsoft... 12/27/2015...
2015Turb... 139,483 Adobe A... 4/10/2016 ...
2016-11-... 343,474 Adobe A... 11/29/2016...
.mo... File fol... 1/14/201... drwxr-... french...
publ... File fol... 1/14/201... drwxr-... french...
.bas... 31 BASH_... 1/17/201... -rw----... french...
.bas... 33 BASH_... 1/14/201... -rw-r--... french...
.bas... 141 BASH_... 1/14/201... -rw-r--... french...
.bas... 124 BASHR... 1/14/201... -rw-r--... french...
.em... 515 EMAC... 1/14/201... -rw-r--... french...
72 files and 3 directories. Total size: 2,413,528,245 bytes
6 files and 2 directories. Total size: 1,502 bytes
Server/Local file Dire... Remote file Size Prio... Status
Queued files Failed transfers Successful transfers
Queue: empty

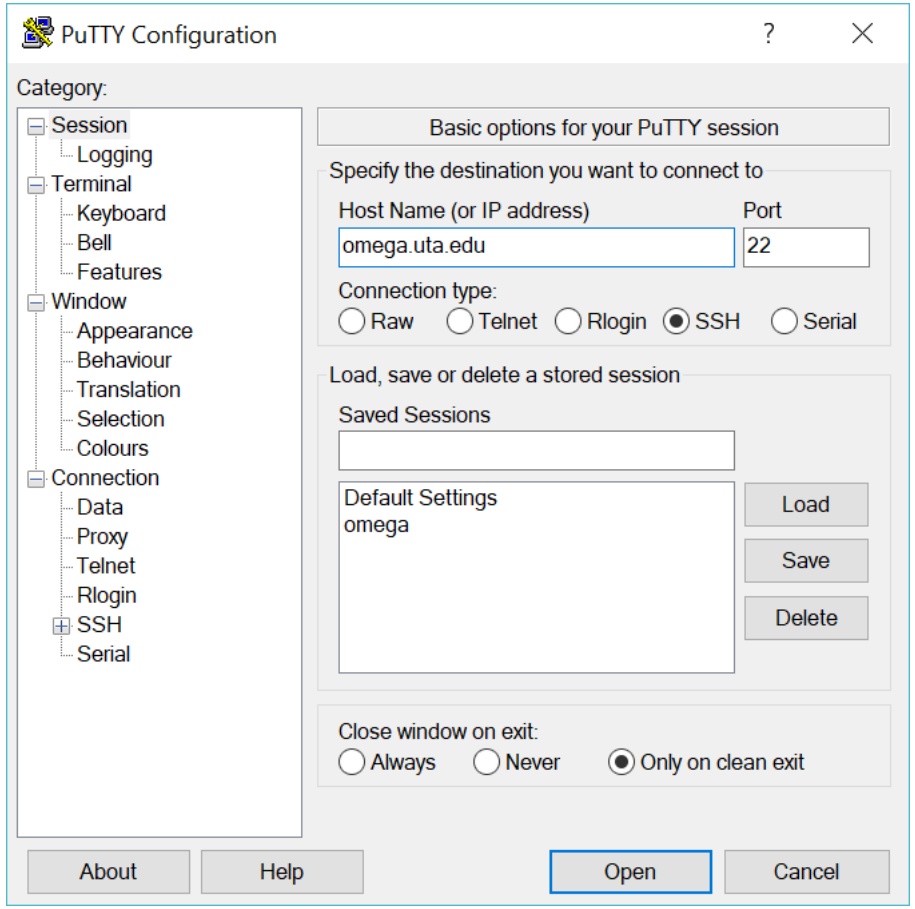
FileZilla

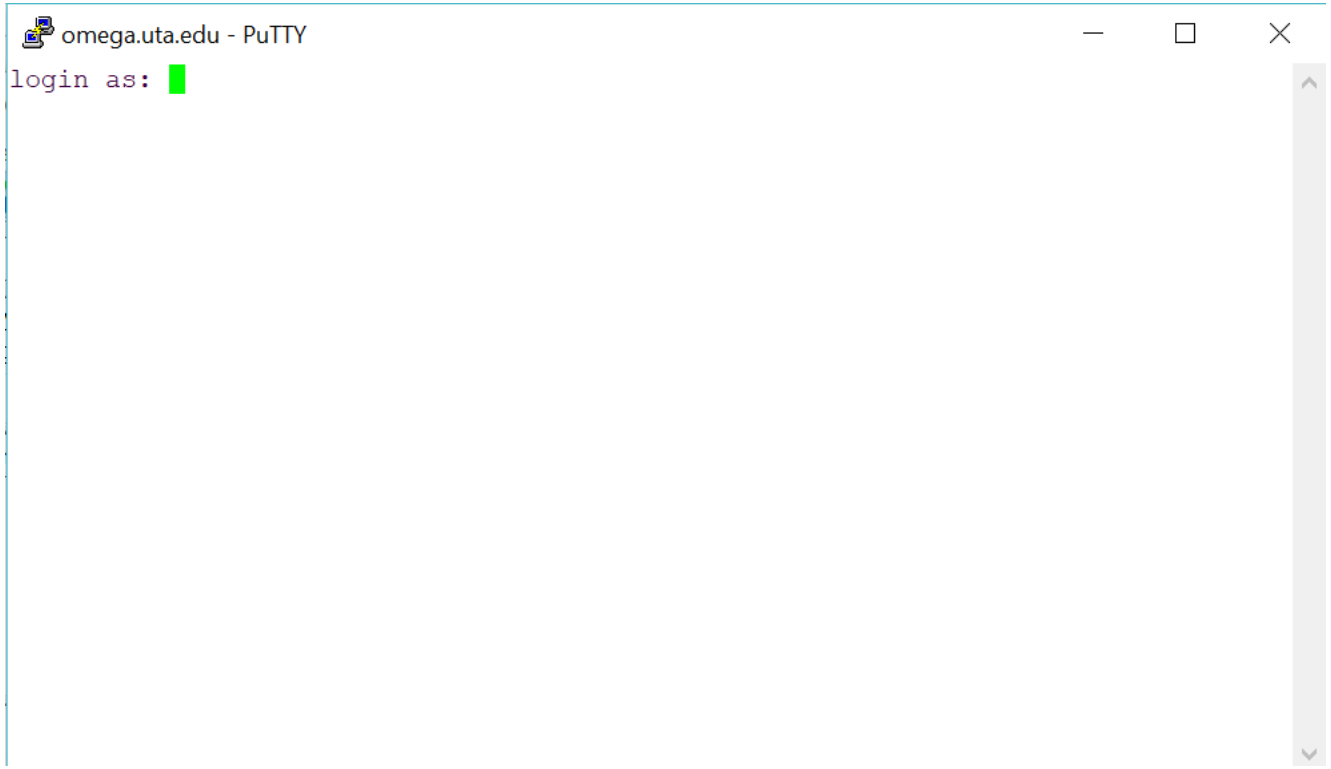


Terminal Emulator

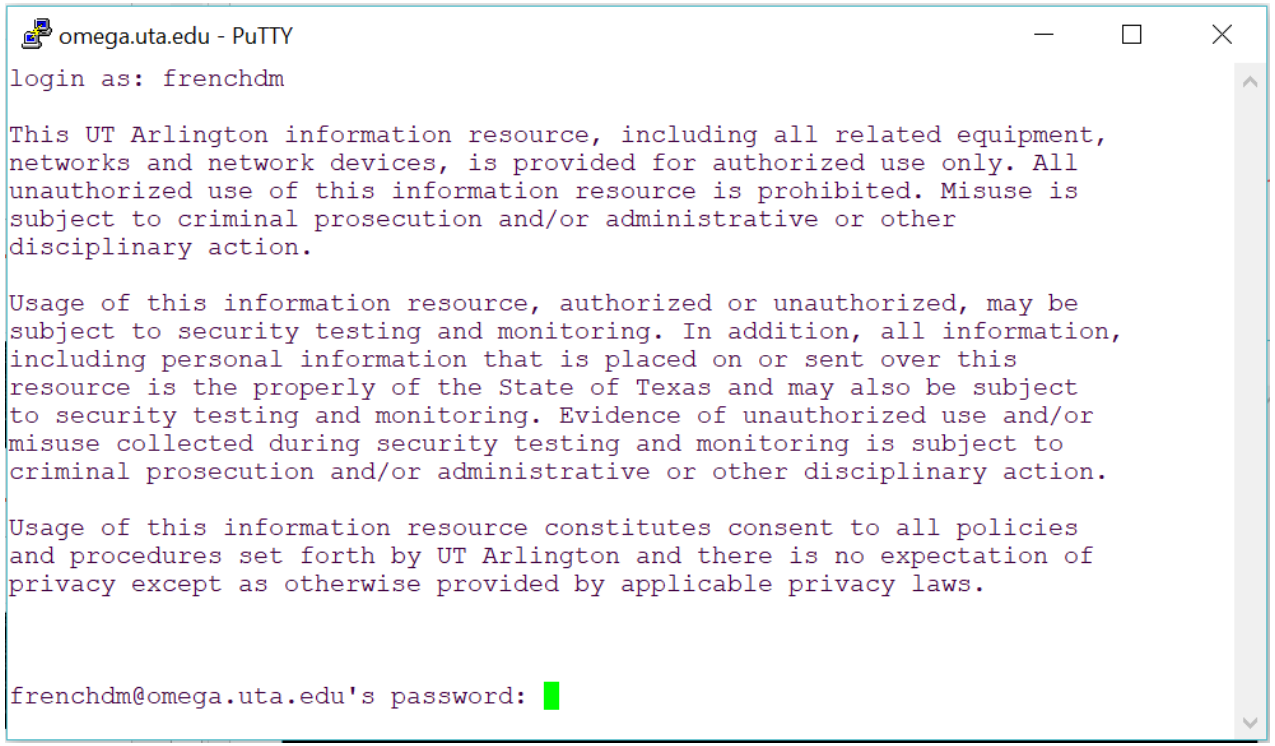


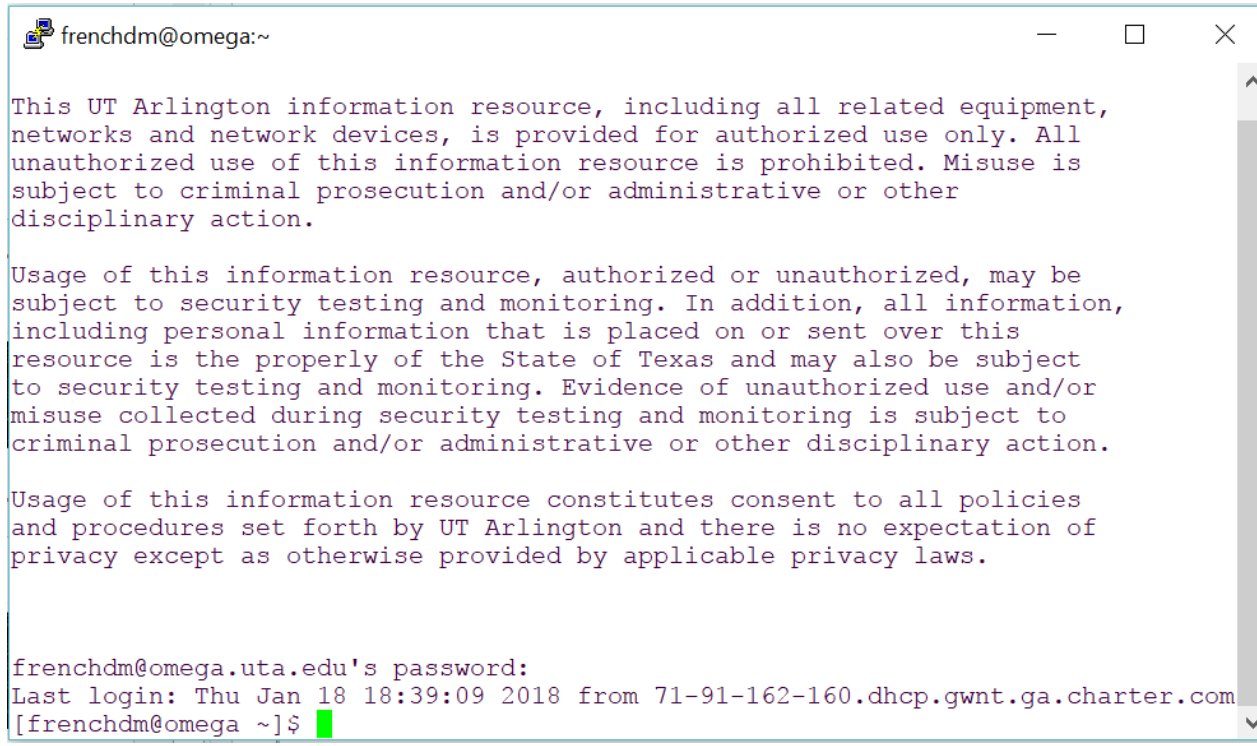
PuTTY


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Serial
Basic options for your PuTTY session
Specify the destination you want to connect to
Host Name (or IP address)
Port
omega.uta.edu
22
Connection type:
Raw
Telnet
Rlogin
SSH
Serial
Load, save or delete a stored session
Saved Sessions
Default Settings
omega
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Help
Open
Cancel

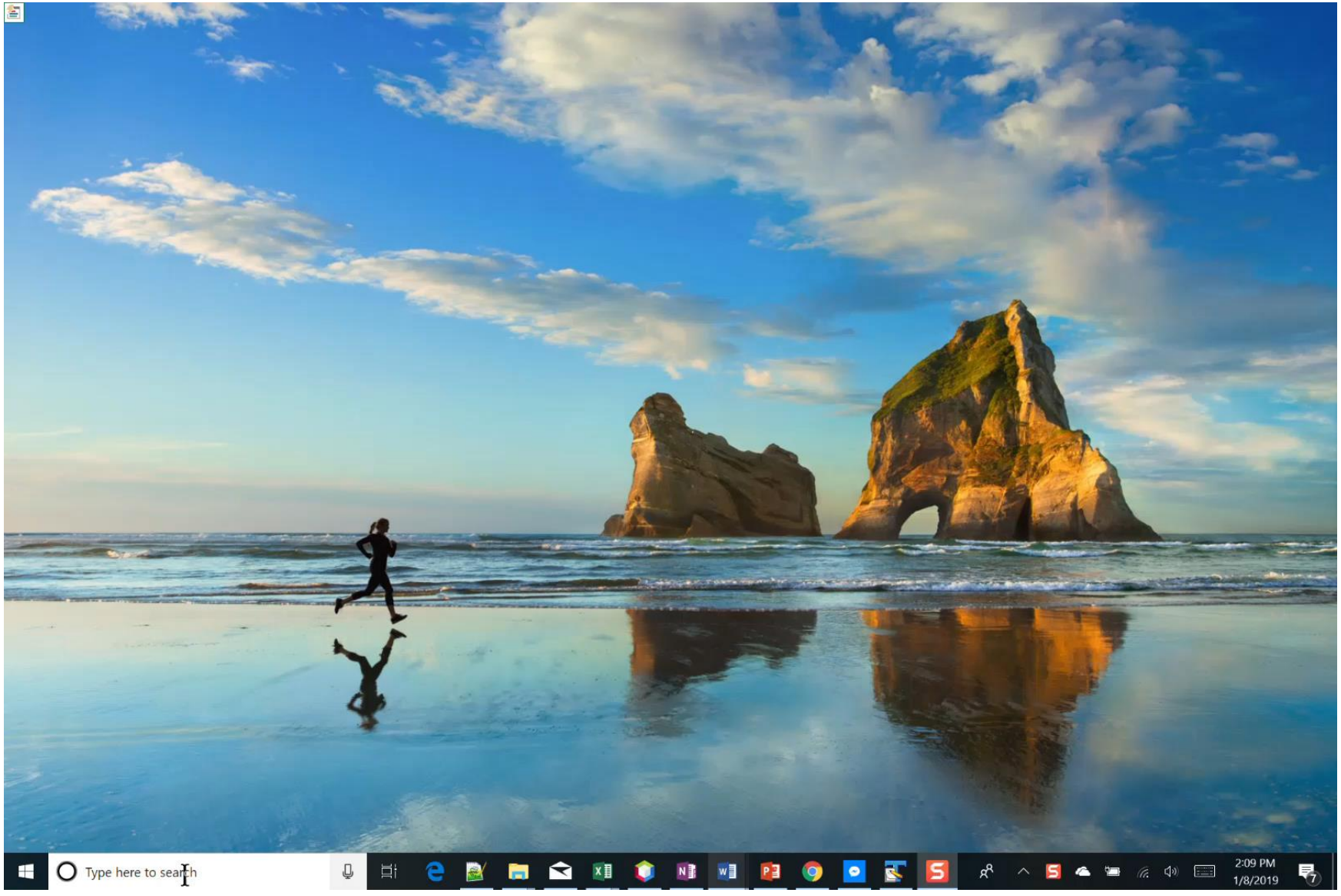

omega.uta.edu - PuTTY
login as:

PuTTY


omega.uta.edu - PuTTY
login as: frenchdm
This UT Arlington information resource, including all related equipment,
networks and network devices, is provided for authorized use only. All
unauthorized use of this information resource is prohibited. Misuse is
subject to criminal prosecution and/or administrative or other
disciplinary action.
Usage of this information resource, authorized or unauthorized, may be
subject to security testing and monitoring. In addition, all information,
including personal information that is placed on or sent over this
resource is the properly of the State of Texas and may also be subject
to security testing and monitoring. Evidence of unauthorized use and/or
misuse collected during security testing and monitoring is subject to
criminal prosecution and/or administrative or other disciplinary action.
Usage of this information resource constitutes consent to all policies
and procedures set forth by UT Arlington and there is no expectation of
privacy except as otherwise provided by applicable privacy laws.
frenchdm@omega.uta.edu's password:

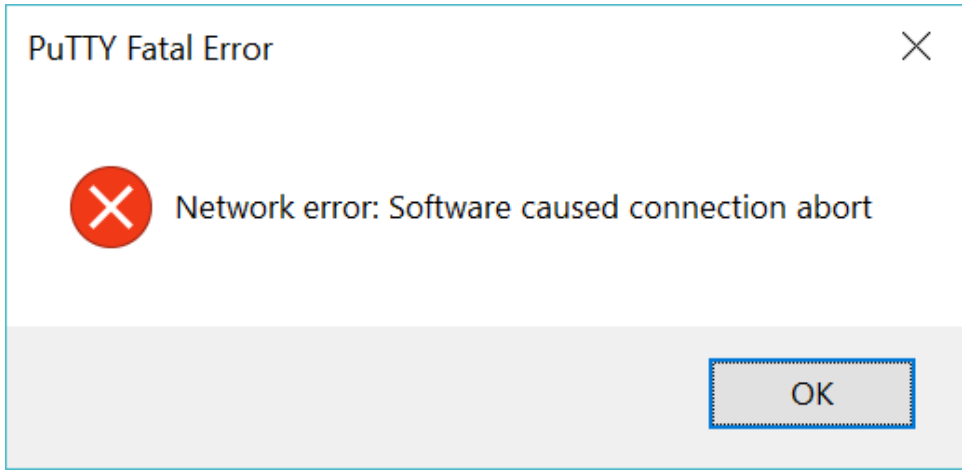

frenchdm@omega:~
This UT Arlington information resource, including all related equipment,
networks and network devices, is provided for authorized use only. All
unauthorized use of this information resource is prohibited. Misuse is
subject to criminal prosecution and/or administrative or other
disciplinary action.
Usage of this information resource, authorized or unauthorized, may be
subject to security testing and monitoring. In addition, all information,
including personal information that is placed on or sent over this
resource is the properly of the State of Texas and may also be subject
to security testing and monitoring. Evidence of unauthorized use and/or
misuse collected during security testing and monitoring is subject to
criminal prosecution and/or administrative or other disciplinary action.
Usage of this information resource constitutes consent to all policies
and procedures set forth by UT Arlington and there is no expectation of
privacy except as otherwise provided by applicable privacy laws.
frenchdm@omega.uta.edu's password:
Last login: Thu Jan 18 18:39:09 2018 from 71-91-162-160.dhcp.gwnt.ga.charter.com
[frenchdm@omega ~]$

PuTTY

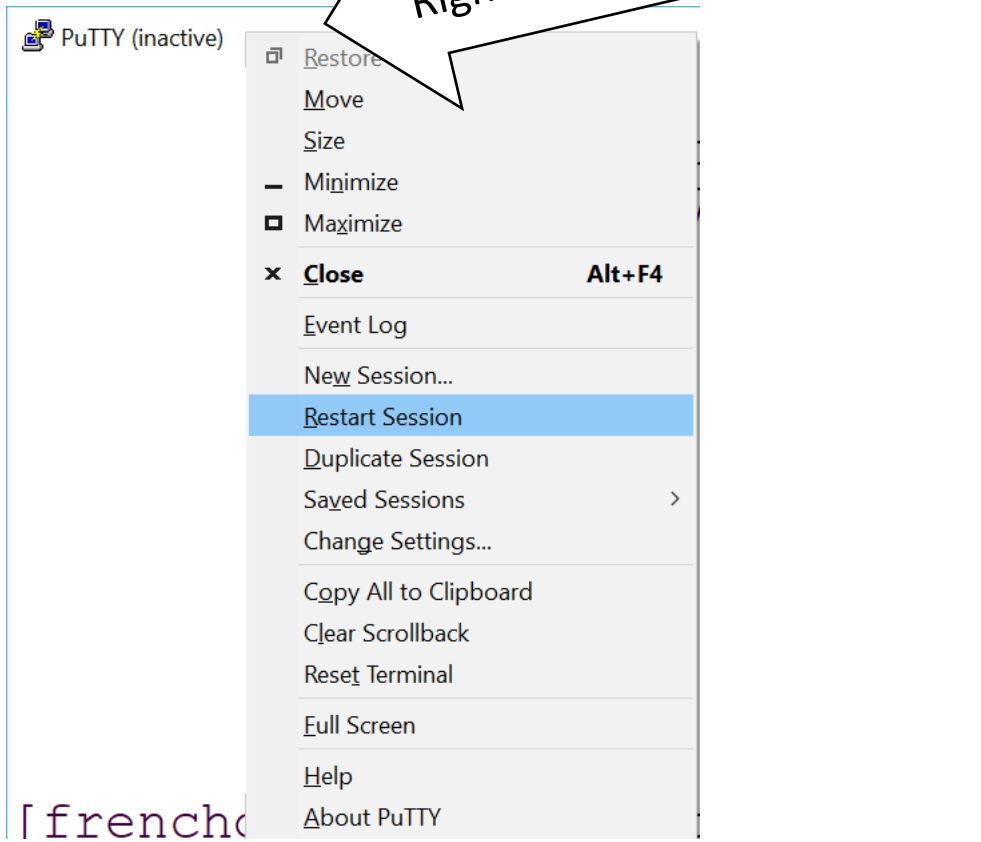


PuTTY

If you leave an Omega session idle for too long, it will automatically disconnect.



When this happens, just restart your current session. No need to quit completely



If you try to SSH to Omega and get either of the following messages

Unable to negotiate….

no matching key exchange method found

no matching cipher found

```
C:\Users\      />ssh       @omega.uta.edu
Unable to negotiate with 129.107.56.23 port 22: no matching key exchange method found. Their offer: diffie-hellman-group
-exchange-sha1,diffie-hellman-group14-sha1,diffie-hellman-group1-sha1
```

```
Unable to negotiate with 129.107.56.23 port 22: no matching cipher found. Their offer: aes128-ctr,aes192-ctr,aes256-ctr
```

Type the following at your command prompt

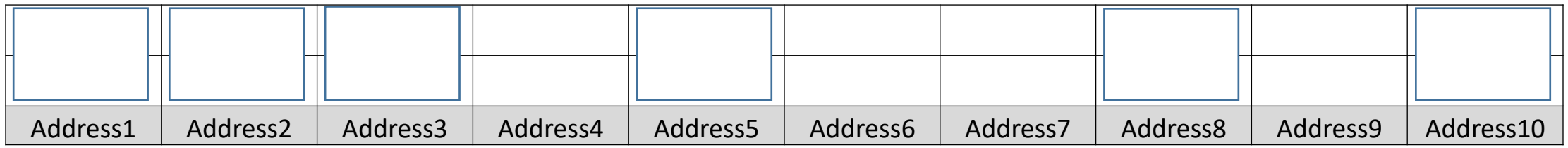
```
ssh -c aes128-ctr -oKexAlgorithms=+diffie-hellman-group1-sha1 yournetid@omega.uta.edu
```

Fill in `yournetid` with your UTA net id.

Pointer Review

- Every variable has an address in memory

```
int VarA = 19;
int VarB = 32;
int VarC = 44;
int IntVar1 = 67
int IntVar2 = 23;
int IntVar3 = 66;
```



Pointer Review

- A pointer can hold that address

```
int *PtrVarA = &VarA;
int *PtrVarC = &VarC;
int *PtrIntVar1 = &IntVar1;
```

VarA	VarB	VarC		IntVar3			IntVar2		IntVar1
19	32	44		66			23		67
Address1	Address2	Address3	Address4	Address5	Address6	Address7	Address8	Address9	Address10

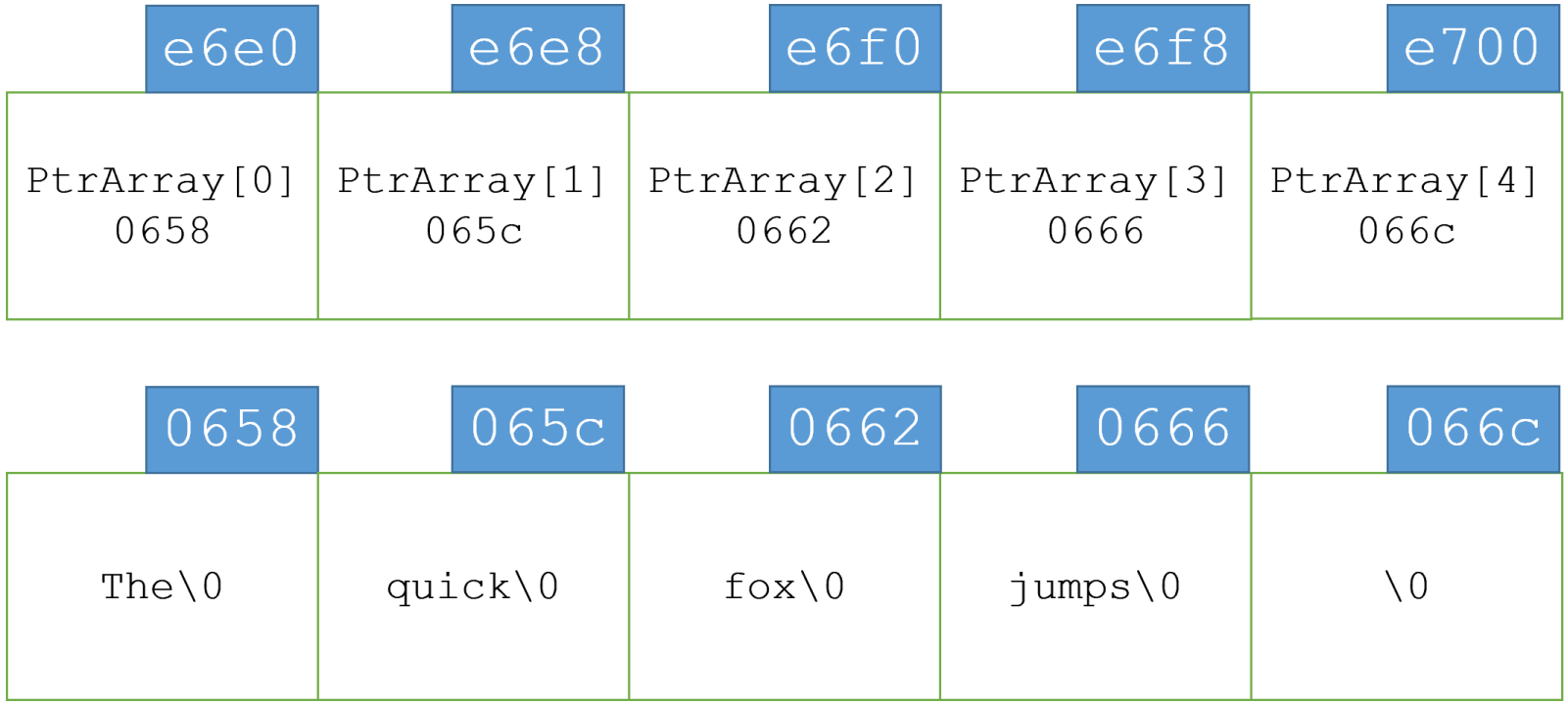
Pointer Review

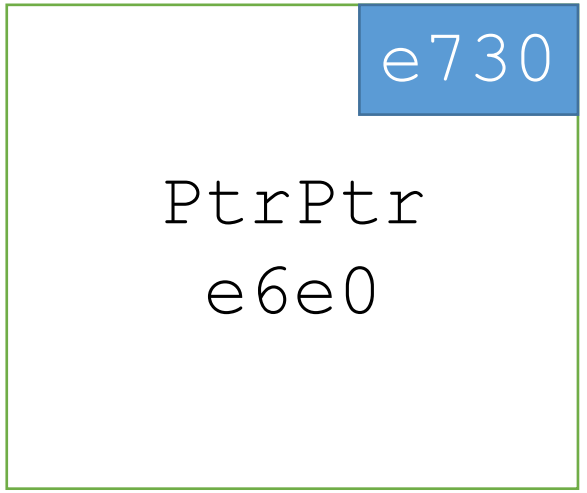
- Dereferencing the pointer gets to the contents

```
printf("Contents of PtrVarA %d", *PtrVarA);
printf("Contents of PtrVarC %d", *PtrVarC);
printf("Contents of PtrIntVar1 %d", *PtrIntVar1);
```

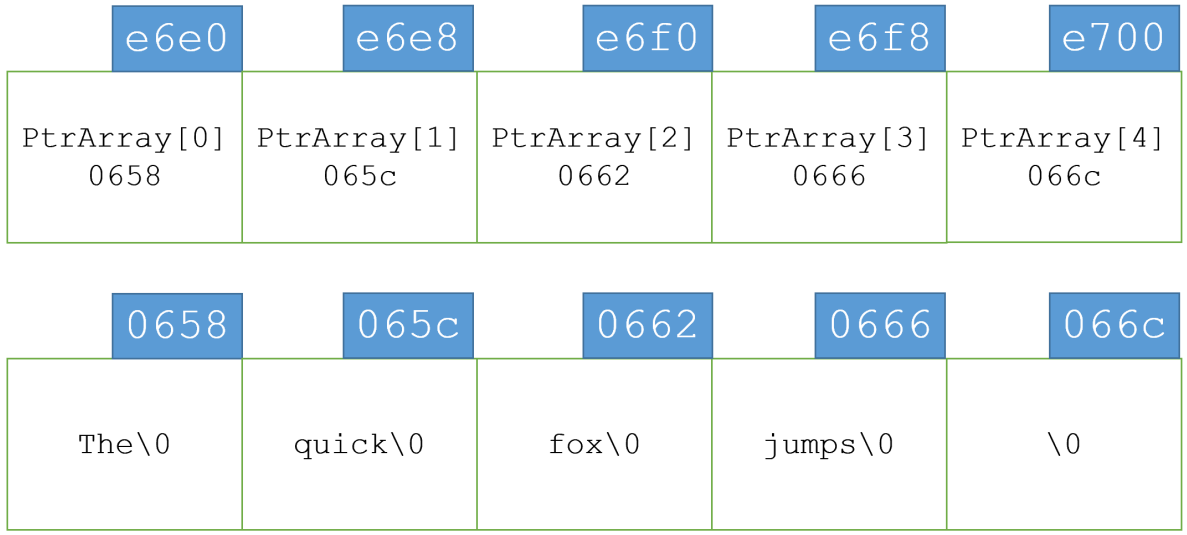
VarA	VarB	VarC	PtrVarA	IntVar3	PtrVarC	PtrIntVar1	IntVar2		IntVar1
19	32	44	Address1	66	Address3	Address10	23		67
Address1	Address2	Address3	Address4	Address5	Address6	Address7	Address8	Address9	Address10

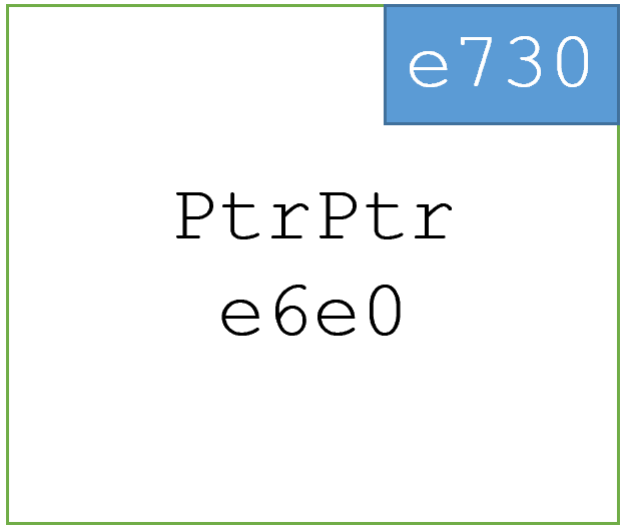
```
char *PtrArray[] = {"The","quick","fox","jumps",""};
char **PtrPtr = PtrArray;
```





```
char *PtrArray[] = {"The","quick","fox","jumps",""};
char **PtrPtr = PtrArray;
```






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

Pointers

```
short *shortVarPtr = NULL;
int   *intVarPtr = NULL;
long  *longVarPtr = NULL;
char  *charVarPtr = NULL;
```

Pointers hold an address and all addresses are the same size

```
printf("The sizeof(short)     is %d\n", sizeof(short));
printf("The sizeof(int)       is %d\n", sizeof(int));
printf("The sizeof(long)      is %d\n", sizeof(long));
printf("The sizeof(char)      is %d\n", sizeof(char));
```

```
The sizeof(short)     is 2
The sizeof(int)       is 4
The sizeof(long)      is 8
The sizeof(char)      is 1
```

```
printf("The sizeof(shortVarPtr) is %d\n", sizeof(shortVarPtr));
printf("The sizeof(intVarPtr)   is %d\n", sizeof(intVarPtr));
printf("The sizeof(longVarPtr)  is %d\n", sizeof(longVarPtr));
printf("The sizeof(charVarPtr)  is %d\n\n", sizeof(charVarPtr));
```

```
The sizeof(shortVarPtr) is 8
The sizeof(intVarPtr)   is 8
The sizeof(longVarPtr)  is 8
The sizeof(charVarPtr)  is 8
```

Pointer Arithmetic

A pointer may be incremented (++) or decremented (--)

```
IntVarPtr++          ++IntVarPtr
IntVarPtr--          --IntVarPtr
```

An integer may be added to a pointer or subtracted from a pointer

```
IntVarPtr += 2   IntVarPtr = IntVarPtr - 45
```

One pointer may be subtracted from another of the same type

```
IntVarPtr1 = IntVarPtr2 - IntVarPtr3
```

The amount of the increment/decrement is relative to the `sizeof()` the type the pointer is pointing to.

```c
#include <stdio.h>

#define MAX_CELLS 10

int main(void)
{
  int  *IntVarPtr = NULL;
  int  IntArray[MAX_CELLS] = {134,278,312,467,523,687,789,811,987,101};
  int i;

  IntVarPtr = IntArray;

  for (i = 0; i < MAX_CELLS; i++)
  {
    printf("IntArray[%d] = %d\t", i, IntArray[i]);
    printf("IntArrayPtr + %d = %d\t", i, *(IntVarPtr + i));
    printf("IntArray + %d = %d\n", i, *(IntArray + i));
  }

  return 0;
}
```

ptrarith1Demo.c

```
IntVarPtr = IntArray;

for (i = 0; i < MAX_CELLS; i++)
{
  printf("IntArray[%d] = %d\t", i, IntArray[i]);
  printf("IntArrayPtr + %d = %d\t", i, *(IntVarPtr + i));
  printf("IntArray + %d = %d\n", i, *(IntArray + i));
}
```

```
IntArray[0] = 134
IntArray[1] = 278
IntArray[2] = 312
IntArray[3] = 467
IntArray[4] = 523
IntArray[5] = 687
IntArray[6] = 789
IntArray[7] = 811
IntArray[8] = 987
IntArray[9] = 101
```

```
IntArrayPtr + 0 = 134
IntArrayPtr + 1 = 278
IntArrayPtr + 2 = 312
IntArrayPtr + 3 = 467
IntArrayPtr + 4 = 523
IntArrayPtr + 5 = 687
IntArrayPtr + 6 = 789
IntArrayPtr + 7 = 811
IntArrayPtr + 8 = 987
IntArrayPtr + 9 = 101
```

```
IntArray + 0 = 134
IntArray + 1 = 278
IntArray + 2 = 312
IntArray + 3 = 467
IntArray + 4 = 523
IntArray + 5 = 687
IntArray + 6 = 789
IntArray + 7 = 811
IntArray + 8 = 987
IntArray + 9 = 101
```

ptrarith1Demo.c

```
for (i = 0; i < MAX_CELLS; i++)
{

  printf("IntArrayPtr + %d = %d\t", i, *(IntVarPtr + i));

}
```

```
for (i = 0; i < MAX_CELLS; i++, IntVarPtr++)
{

  printf("IntArrayPtr + %d = %d\t", i, *IntVarPtr);

}
```

Difference between

`*IntVarPtr + i` `*(IntVarPtr + i)`

ptrarith2Demo.c

```
  for (i = 0; i < MAX_CELLS; i++, CharVarPtr++)
  {
    printf("CharArray[%d] = %c CharVarPtr = %p *CharVarPtr = %c\n",
           i, CharArray[i], CharVarPtr, *CharVarPtr);
  }
  for (i = 0; i < MAX_CELLS; i++, IntVarPtr++)
  {
    printf("IntArray[%d] = %d IntVarPtr = %p *IntVarPtr = %d\n“,
           i, IntArray[i], IntVarPtr, *IntVarPtr);
  }
  for (i = 0; i < MAX_CELLS; i++, LongVarPtr++)
  {
    printf("LongArray[%d] = %ld LongVarPtr = %p *LongVarPtr = %d\n",
           i, LongArray[i], LongVarPtr, *LongVarPtr);
  }
```

```
CharArray
{"ABC"}

IntArray
{134,278,312}

LongArray
{111,222,333}
```

```
CharArray[0] = A CharVarPtr = 0x7fff4d0170c0 *CharVarPtr = A
CharArray[1] = B CharVarPtr = 0x7fff4d0170c1 *CharVarPtr = B
CharArray[2] = C CharVarPtr = 0x7fff4d0170c2 *CharVarPtr = C

IntArray[0] = 134 IntVarPtr = 0x7fff4d0170d0 *IntVarPtr = 134
IntArray[1] = 278 IntVarPtr = 0x7fff4d0170d4 *IntVarPtr = 278
IntArray[2] = 312 IntVarPtr = 0x7fff4d0170d8 *IntVarPtr = 312

LongArray[0] = 111 LongVarPtr = 0x7fff4d0170a0 *LongVarPtr = 111
LongArray[1] = 222 LongVarPtr = 0x7fff4d0170a8 *LongVarPtr = 222
LongArray[2] = 333 LongVarPtr = 0x7fff4d0170b0 *LongVarPtr = 333
```

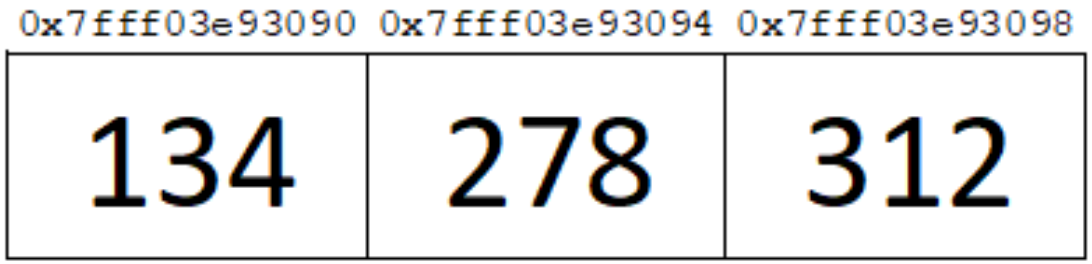
Pointer arithmetic works for all different types.

ptrarith3Demo.c

```
int  IntArray[MAX_CELLS] = {134,278,312};
int  *IntVarPtr1 = IntArray;
int  *IntVarPtr2 = IntArray+1;
int  *IntVarPtr3 = IntVarPtr2+1;

printf("*IntVarPtr1 = %d\n", *IntVarPtr1);
printf("*IntVarPtr2 = %d\n", *IntVarPtr2);
printf("*IntVarPtr3 = %d\n", *IntVarPtr3);

printf("IntVarPtr3 - IntVarPtr1 = %d\n", IntVarPtr3 - IntVarPtr1);
printf("IntVarPtr1 - IntVarPtr3 = %d\n", IntVarPtr1 - IntVarPtr3);
```




```
*IntVarPtr1 = 134
*IntVarPtr2 = 278
*IntVarPtr3 = 312

IntVarPtr3 - IntVarPtr1 = 2
IntVarPtr1 - IntVarPtr3 = -2
```

Represents the distance between them

If I take the physical address of one house and “subtract” the physical address of a house down the street, then I would get the number of houses in between them.

If I take Setember 30th and “subtract” September 12th, then I would get the number of days in between them.

Pointer Arithmetic

Allowed operations

- A pointer may be incremented (++) or decremented (--)
- An integer may be added to a pointer or subtracted from a pointer
- One pointer may be subtracted from another of the same type

What about Pointer Addition?

```
printf("IntVarPtr1 + IntVarPtr2 = %d\n", IntVarPtr1 + IntVarPtr2);

[frenchdm@omega ~]$ gcc ptrarith4Demo.c
ptrarith4Demo.c: In function ‘main’:
ptrarith4Demo.c:23: error: invalid operands to binary +
```

- Not defined in the language. What would it mean?
- You can subtract two dates to get the number of days in between them. What would adding two dates mean?

Pointer Arithmetic

```
int Array1 = {1,2,3}
int Array2 = {4,5,6}

Array2[0] != Array1 + 3
```

Allowed operations

- A pointer may be incremented (++) or decremented (--)
- An integer may be added to a pointer or subtracted from a pointer

Pointer arithmetic is only used within arrays where the order of cells in memory is guaranteed.

Pointer arithmetic should not be used to travel between arrays.

Adding to/subtracting from a pointer does not guarantee the next/previous variable in your list of declarations – memory is not necessarily arranged in the order of your declarations.

Rubber Duck Debugging

https://en.wikipedia.org/wiki/Rubber_duck_debugging

In software engineering, rubber duck debugging is a method of debugging code.

The name is a reference to a story in the book "The Pragmatic Programmer" in which a programmer would carry around a rubber duck and debug their code by forcing themselves to explain it, line-by-line, to the duck.

Many other terms exist for this technique, often involving different (usually) inanimate objects (teddy bear) or pets such as a dog or a cat.

Rubber Duck Debugging

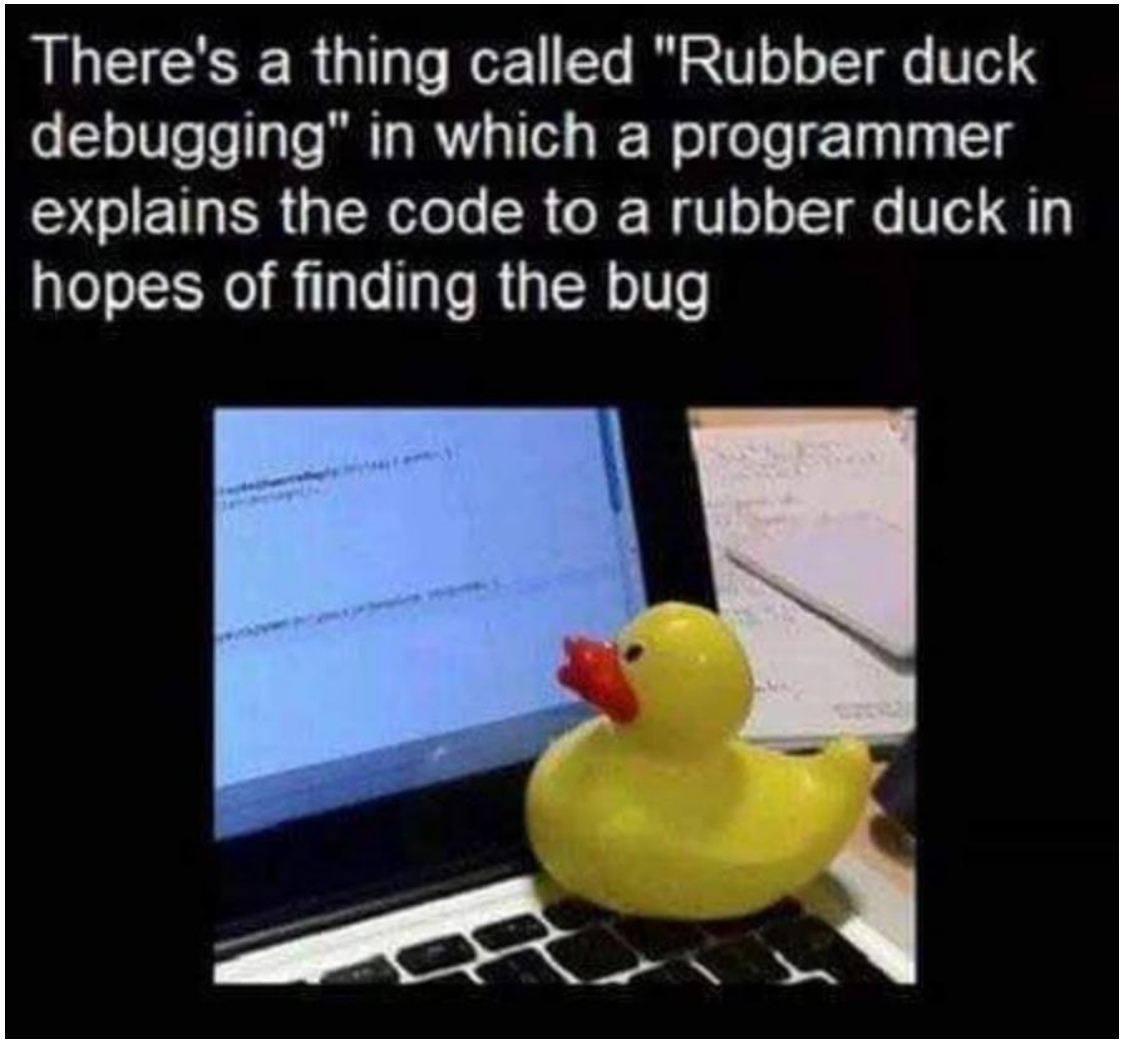
Many programmers have had the experience of explaining a problem to someone else, possibly even to someone who knows nothing about programming, and then hitting upon the solution in the process of explaining the problem.

In describing what the code is supposed to do and observing what it actually does, any incongruity between these two becomes apparent.

More generally, teaching a subject forces its evaluation from different perspectives and can provide a deeper understanding.

By using an inanimate object, the programmer can try to accomplish this without having to interrupt anyone else.

imgur explaining the problem solves half the problem



I work at a startup and part of the onboarding package you get when you first start working here now includes a rubber duck. We also have a bigger version of the duck for the extra hard problems. Sometimes one duck doesn't cut it and you need to borrow your neighbors to get more ducks on the problem. One time we couldn't figure out why something wasn't working right so we assembled the counsel of ducks and by the grace of the Duck Gods were we able to finally come to a solution. These ducks have saved many lives and should be respected for the heroes they are.

imgur **explaining the problem solves half the problem**

Some of you are reblogging because you think its funny that programmers would talk to ducks. I'm reblogging because I think its funny picturing a programmer explaining their code, realizing what they did when they explain the bad code, then grabbing the strangling the duck while yelling "WHY WAS THE FIX THAT SIMPLE!? AM I GOING BLIND!"

AS A PROGRAMMER I CAN TELL YOU THAT THIS IS EXACTLY WHAT YOU [redacted] DO WE HAD TO BAN THE DUCKS FROM MY CLASSES BECAUSE EVERYONE WOULD FLIP THE DUCK OR THROW IT AT A WALL OR SOMETHING WHEN THEY FIGURED OUT THE PROBLEM IN THEIR CODE

Action Items

Mon, Mar 27

Due 11:59pm Homework 5

Due 11:59pm Coding Assignment 4

Due 11:59pm Crash Course : Quiz 8

Wed, Mar 29

Due 11:59pm OLQ9