EEB 177 Lecture 3

Office Hours

Tues 2-3 Wednesday 1-2 Terasaki 2149

Preliminaries

- Start **nano** and save the file "classwork-Thursday-1-16.txt" to your class-assignments directory
- push this to your remote repository
- you can write answers to today's exercises in this file.

Topics

- shell and scripting

absolute and relative paths

- the absolute path is the entire path starting from the root, /
  - `/Users/michael_alfaro/tools`
- the relative path is defined relative to the current directory
  - if we were in the/`tools` directory above, we could get to the root using the relative path: `cd ../../../`
  -

    we could also go there using the absolute path: `cd /`

cp

- copies files
- `cp file_to_copy path_to_destination`

```
# If you specify the full path,
# your current location does not matter
$ cp ~/CSB/unix/data/Buzzard2015_about.txt
    ~/CSB/unix/sandbox/
# assume your current location is the unix sandbox
# we can use a relative path
$ cp ../data/Buzzard2015_about.txt .
# the dot is shorthand to say "here"
# rename the file in the copying process
cp ../data/Buzzard2015_about.txt ~/buzz-2015.txt
```

mv

moves (or renames) files

- `mv file_to_move destination`

```
# move the file to the data directory

    `$ mv Buzzard2015_about2.txt ../data/`

# rename a file
$ mv ../data/Buzzard2015_about2.txt Buzzard-2015.txt
```

touch

- creates an empty file (or updates date of access of exiting file)
- `touch my_file_name.txt`

rm

- removes a file
- `rm -r` removes files recursively

mkdir

- creates an empty directory
- `mkdir my_new_dir`
- to make nested directories use `-p`
- `mkdir -p dir_a/dir_b/dir_c`

rmdir and rm

- removes an empty directory
- `rmdir directory_name`
- be careful–no trashcan in unix!
- use `rm -r` to recursively remove files from non-empty directory
- `rm -r directoryname`

history

- shows the command line history from the current session (this is a BASH command)
- `history n` (try it)

>

the > symbol redirects output. This turns out tobe very handy. We can redirect the command line history, for example, like so:

```
history n > unix-commands-used-in-class.txt
```

try it! Check to see if this file shows up in your current directory.

Challenge 3

1. cd to your class exercise directory
2. Create a nested directory sequence that corresponds to Linnean taxonomy hierarchy in one line.
3. Navigate to the species directory and create three text files corresponding to Disney animal sidekicks.
4. Move one file to the Class directory using the aboslute path.
5. Copy a second file to the Family directory using the relative path
6. Delete the genus directory.
7. Push your command history file to the remote repo

Printing and modulating files

less

progressively print file to screen `$ less Marra2014_data.fasta`

>contig00001 length=527 numreads=2 gene=isogroup00001
status=it_thresh

ATCCTAGCTACTCTGGAGACTGAGGATTGAAGTTCAAAGTCAGC

...‘

cat

concatenate and print files to screen `cat Marra2014_about.txt Gesquiere2011_about.txt Buzzard2015_about.txt --`

wc

line, word, and character count of a file

`wc Gesquiere2011_about.txt` >8 64 447 ../data/Gesquiere2011_about.txt

challenge

a) Go to the data directory within CSB/unix

(b) How many lines are in file Marra2014_data.fasta?

(c) Create the empty file toremove.txt in the CSB/unix/sandbox di- rectory without leaving the current directory.

(d) List the contents of the directory unix/sandbox.

(e) Remove the file to remove.txt.

Wildcards

You used wildcards in your lab. In the shell we can use the `*` wildcard (match 0 or more characters except for a leading `.`) to find specific file types.

For example, to see only text files we could type: `ls *.txt`. To see the beginning of all files that start with pp we could type `head -n 2 pp*`.

see 1.6.4 in text for more examples

less

this command lets you examine the contents of large text files. You can move through these pages with `ctrl-f` and `ctrl-b`. Exit `less` with `q`; `h` for more commands.

Try this

examine the file

`Marra2014_data.fasta` in the `/CSB/unix/data directory`

wc

gives line, word, and byte count of a file. Look at the `-w -l -c -m` options in the manual. What do they do?

Try this

how man words in are the file `Marra2014_about.txt` ?

sort

sorts lines of a file alphabetically or numerically (with `-n`). To choose a specific column for sorting, choose `-k`. For revese sort use `-r`.

Try this

numerically sort the file `Gesquiere2011_data.csv` by the second column.

head and tail

these commands show the beginning and end of a text file. Use `-n` to specify the number of lines to show.

Try this

show the first and last five lines of the file `Gesquiere2011_data.csv`

show everything but the first line of `Gesquiere2011_data.csv` (hint: see the manual on how to start from a specific line)

appending to a file

`>>` *appends* output to a file.

```
ls > current_dir_contents.txt
```

```
cat history >> myhomework.txt
```

#challenge
The `echo` command prints a string to the screen. Tell the shell to print your name. Now tell the shell to print your name to a file called `name.txt`

pipe and redirect example

Lets use the commands you have learned already to avoid a tedious task. Imagine that you want to know the number files within the folder `Saavedra2013`. How could you do this?

pipe and redirect example

One way woud be to go to the folder and then count by hand. But this would be tedious!

pipe and redirect example

We can use the shell to do this in two steps by creating a text file that contains all of the file names and then counting the length of that file.

```
ls ../data/Saavedra2013 >> filelist.txt wc -l
filelist.txt
```

But we can do even better using the pipe command, `|`. Pipe says take the output on the left side and send it as input to the right side. So, for example, we can do the above in one line:

```
ls ../data/Saavedra2013 | wc-l
```

csv files

One of the most common and useful formats for tabular data is **.csv** (Comma Separated Values) where columns are separated by a comma or other delimiter.

Bolstad2015_data.csv

```
Species,ID,Date,Sex,WingSize,L2Length
D_acutila,ACU1006.TIF,24_Jul_01,F,0.1311220183,0.4972620127
D_acutila,ACU1009.TIF,24_Jul_01,F,0.1360880957,0.4879716426
D_acutila,ACU1010.TIF,24_Jul_01,F,0.1953932712,0.5401365988
D_acutila,ACU1013.TIF,24_Jul_01,F,0.2773285363,0.6463595018
D_acutila,ACU1018.TIF,24_Jul_01,F,0.1515311551,0.4977579266
D_acutila,ACU1021.TIF,24_Jul_01,F,0.1751864614,0.4919342481
D_acutila,ACU1048.TIF,24_Jul_01,F,0.2297430676,0.600350886
D_acutila,ACU1049.TIF,24_Jul_01,F,0.1744773274,0.5457801765
D_acutila,ACU1054.TIF,05_Sep_01,F,0.1080136588,0.4291605164
D_acutila,ACU1059.TIF,05_Sep_01,F,0.2047006161,0.5799392151
D_acutila,ACU1060.TIF,05_Sep_01,F,0.1905922086,0.5332696783
D_acutila,ACU1061.TIF,05_Sep_01,F,0.265253513,0.5802409956
D_acutila,ACU1062.TIF,05_Sep_01,F,0.1938309097,0.5197846926
D_acutila,ACU1063.TIF,05_Sep_01,F,0.2937455662,0.6506255569
D_acutila,ACU1064.TIF,05_Sep_01,F,0.2274851883,0.5594804043
D_acutila,ACU1078.TIF,05_Sep_01,F,0.2068273804,0.5517276433
D_acutila,ACU1089.TIF,05_Sep_01,F,0.2645185555,0.6127634599
D_acutila,ACU1090.TIF,05_Sep_01,F,0.1788871209,0.4985366955
```

working with csv files in the shell

We can use several commands you have learned already plus the `cut` command to easily manipulate csv files.

First, take a look at `Pacifici2013_data.csv` using your text editor. Then move to the containing diretory and use a unix command to view the first line (only) of that file.

What is the delimiter in this file?

```
head -n 1 Pacifici2013_data.csv
```

We can use cutto extract specific fields by specifying the delimiter with `-d` and the desired columns with `-f`argument.

```
head -n 1 Pacifici2013_data.csv | cut -d ';' -f 1-4
```

If we wanted to list rows of data without the header, we can pipe the results of cut to `tail` (remember `tail -n +2` will show the contents of a file or stream starting from the second line.

```
cut -d ';' -f 2 | head -n 5| tail -n +2
```

Challenge

Show the Order of the first 5 species in the data set. Append this to your class-exercises files for today.
(hint: you will need `cut`, `|`, `tail`, and `head`)

Challenge

use what you know plus the `uniq` command to count the number of unique families in this file.**hint:** you will need to sort your data before you apply `uniq`. Append the line "There are X unique families:" (fill in the value for x) to your exercise file. Then append the list of unique families to your exercise file.

Reformatting a csv file

We will now work through example 1.6.2 to create a data file with the following fields: Order, Family, Genus, Scientific_Name and AdultBodyMass_g with the following properties

- no headers
- data are sorted by size from large to small
- delimiter is a space

We will need to introduce the `tr` command to translate characters.

Preliminaries

- Start **nano**: `$ nano` and save the file "classwork-Thursday-1-16.txt" to your class-assignments directory
- push this to your remote repository
- you can write answers to today's exercises in this file.

grep review

. Grep is a powerful pattern matching command that can be combined with the regular expressions you used in lab.

Useful grep options: - `-c` to count lines - `-w` to match words - `-i` to make case insensitive - `-n` to show line number of match.

use grep to find all of the protected orchids (Orchidaceae) in this list

use grep to find all of the protected orchids (Orchidaceae) in this list

```
grep Orchidaceae cep-taxa.txt
```

how could you count these lines?

use grep to find all of the protected orchids (Orchidaceae) in this list

```
grep Orchidaceae cep-taxa.txt
```

how could you count these lines? `grep -c Orchidaceae cep-taxa.txt`

use grep to find all of the falcons in this list.

try searching for `falcon`. Then try searching for `Falcon`. What is going on? How can you find occurrences regardless of case?

use grep to find all of the falcons in this list.

try searching for `falcon`. Then try searching for `Falcon`. What is going on? How can you find occurrences regardless of case?

```
grep -i falcon cep-taxa.txt
```

Macaws are in the genus Ara. Find all of the macaws in this list regardless of case. What is the problem now?

Macaws are in the genus Ara. Find all of the macaws in this list regardless of case. What is the problem now?

How can you solve it?

Macaws are in the genus Ara. Find all of the macaws in this list regardless of case. What is the problem now?

How can you solve it?

```
grep -i -w ara cep-taxa.txt
```

Other grep options

`-B X` finds x lines before `-A X` finds X lines after

find the 3 lines before all occurrences of Cebus.

Other grep options

`-n` shows the line number of the match.

what are the line numbers of all iguanas in the file?

Other grep options

`-n` shows the line number of the match.

what are the line numbers of all iguanas in the file?

```
grep -i -n -w iguana cep-taxa.txt
```

finding all lines that do not match

`-v` returns everything that does not match the grep pattern

finding all lines that do not match

`-v` returns everything that does not match the grep pattern

How many lines do not match ara?

what are the line numbers of those lines?

finding all lines that do not match

`-v` returns everything that does not match the grep pattern

How many lines do not match ara?

```
grep -i -v -c ara cep-taxa.txt
```

what are the line numbers of those lines? `grep -i -v -n ara cep-taxa.txt`

finding files with `find`

`find` allows you to search for **files** with specified attributes.

use the wildcard `.*` to find everything in your sandbox directory.

```
find .
```

```
.
./temp.txt
./cep-taxa.txt
./.gitignore
./junk
...
```

if you pass find a path it will give all files and folders in that directory

```
find ../data/
```

```
../data
../data/toremove.txt
../data/Gesquiere2011_data.csv
../data/Saavedra2013_about.txt
...
```

if you pass find a path it will give all files and folders in that directory

```
find ../data/
```

```
../data
../data/toremove.txt
../data/Gesquiere2011_data.csv
../data/Saavedra2013_about.txt
...
```

count all of the files and folders within `../data/`

`find` options

You can search for a specific file with `-name`

```
find  ../data/ -name "n30.txt"
```

This can be helpful when you don't know exactly where you left a file.

```
find /home/eeb-177-student/Desktop/ -name
"classwork-Tues-1-17.txt"
```

```
/home/eeb-177-student/Desktop/eeb-177/class-assignments/cla
```

`find` gets even more powerful with wildcards.

for example, to find all of the files with `about` in the data directory….

```
find /home/eeb-177-student/Desktop/eeb-177/CSB/unix/
-iname "*about*"
```

```
/home/eeb-177-student/Desktop/eeb-177/CSB/unix/data/Saavedr
/home/eeb-177-student/Desktop/eeb-177/CSB/unix/data/Marra2(
/home/eeb-177-student/Desktop/eeb-177/CSB/unix/data/Pacific
```

note that `-iname` ignores the case in the file names

find the path to all of your classwork files and append these to you classwork file for today.

find the path to all of your classwork files and append these to you classwork file for today.

```
find /home/eeb-177-student/Desktop/ -iname "*class*"
>>
/home/eeb-177-student/Desktop/eeb-177/class-assignments/cla
```

specifying the depth of the search

to restrict the depth in the folder hierarchy of the search, use the `-maxdepth N` option.

What will this line do?

```
$ find ../data -maxdepth 1 -name  "*.txt"  | wc -l
```

How many text files are there in `../data`if you do not restrict the depth?

You can exclude certain files with `not`

```
find ../data/ -not -name "*about*" | wc -l
```

Permissions

In Unix, each file and directory has an attribute that determines who can read (r), write (w), execute (x), or do nothing (-) to a file. There are three categories of file users

- owner
- group (set of users)
- everyone else

you can see permissions with `ls -l`

permissions structure

chmod and chown

These commands change permissions and ownwership (u, g, or o for user, group or other).

```
touch permissions.txt ls -l
```

```
-rw-rw-r-- 1 eeb-177-student eeb-177-student 0 Jan 24 07:51
```

```
chmod u=rwx permissions.txt
ls -l
-rwxrw-r-- 1 eeb-177-student eeb-177-student      0 Jan 24 0
```

notice that the user may now execute this file.

you can also add and remove permissions for a user with + and -.

```
chmod g+w,u+x permissions.txt
ls -l permissions.txt
-rwxrw-rw-
```

Add write permissions for all users.

Remove read, write, and execute permission from others

Writing a shell script

Lets illustrate some ideas about paths, scripts, and permissions by writing a simple shell script. You are going to write a program in your text editor that will execute a series of shell commands that you have already learned.

open up gedit and type the following lines:

```
#! /bin/bash
ls -la
echo "Above are the directory listings for this folder"
pwd
echo "right now it is :
date
```

save this file as **dir.sh**

Paths

there are two standard locations for programs– `/usr/bin` and `/bin`

use `ls` to see what is in them

The shell searches these directories (and others that have been addded to the path) whenever you type a command.

Type `echo $PATH` to see your current path.

`which` will tell you the directory to a command. Try `which cat`

Creating a scripts directory and adding it to the path

we want a single working copy of each program on our machines so we need to make sure the shell searches for our programs....

- go to your home directory
- create a directory called `scripts`
- to add the scripts directory to the path, open the `.bash_profile` file in gedit
- add this line (exactly) `export PATH="$PATH:$HOME/scripts"`
- exit and save

Now we have created a program we would like to run and created a path to the scripts directory. What else do we need to do?

hint: where is `dir.sh` right now?

the shebang (#!)

`#!` is called the shebang–it means that all following contents of script will be sent to the program following the shebang

`#! /bin/bash` sends it all to bash

remember,new scripts are not executable w/o changing permissions

checking permissions

- cd ~/scripts
- check permission with ls -l
- add permission to execute with chmod u+x

try running your program from different directories. Does it work? Why?

exercise

Add your scripts directory to your remote repository. You will need to

- `git init` in your scripts directory
- add your script
- commit your script
- create a remote repo on github
- copy and paste the command lines from the remote repo after you create it.

```
git remote add origin
https://github.com/michaelalfaro/eeb-177-scripts.git
git remote add origin
https://github.com/michaelalfaro/eeb-177-scripts.git
git push -u origin master
```

you will be using your scripts directory throughout the quarter, so make sure this repo is working