

EEB 177 Lecture 4

Topics

- ▶ Advanced shell commands
- ▶ Permissions

Office Hours

Thurs 1:30-2:30

Terasaki 2149

Preliminaries

- ▶ Start **nano**: `$ nano` and save the file “classwork-Thursday-1-23.txt” to your class-assignments directory
- ▶ push this to your remote repository
- ▶ you can write answers to today’s exercises in this file.

tr

tr substitutes or removes specific characters in a text file. We will use this to create a new file with body mass data.

```
echo 'aaaabbbb' | tr 'a' 'b' echo '123456789' | tr 1-5 0 echo  
'aabbccddee' | tr acd 123 echo 'aabbccddee' | tr a-c 1-3 echo  
'aaaaabbbb' | tr -d a bbbb
```

create the bodymass file

remove the header

```
tail -n +2 ../data/Pacifici2013_data.csv
```

take columns 2-6

```
tail -n +2 ../data/Pacifici2013_data.csv | cut -d ";"  
-f 2-6
```

substitute a space for the semicommas

```
'tail -n +2 ../data/Pacifici2013_data.csv | cut -d';' -f 2-6 | tr -s ';' '  
,'
```

-># sorting and outputting the file

sort in reverse order of body mass

```
tail -n +2 ../data/Pacifici2013_data.csv | cut -d ";"  
-f 2-6 | tr -s ";" " " | sort -r -n -k 6
```

create the file BodyM.csv

```
tail -n +2 ../data/Pacifici2013_data.csv | cut -d ";"  
-f 2-6 | tr -s ";" " " | sort -r -n -k 6 > BodyM.csv
```

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 wc to the number of species in BodyM.csv

use `grep` to find all of the wombats (Vombatidae) in this list

```
grep Vombatidae BodyM.csv
```

how could you count these lines?

how could you count these lines? `grep -c Vombatidae`
`BodyM.csv`

use grep to find all of the genus Bos in this list.

try searching for Bos. What is going on?

use the -w command to find whole words

```
grep -w Bos BodyM.csv
```

-i makes the search case insensitive

```
grep -i Bos BodyM.csv
```

Other grep options

-B X finds x lines before -A X finds X lines after
find the 3 lines before all occurrences of Bos.

Other grep options

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

```
grep -n "Gorilla gorilla" BodyM.csv
```

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 species in the genus Gorilla besides Gorilla gorilla?

```
grep Gorilla BodyM.csv | grep -v gorilla
```

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
```

```
...
```

Challenge

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/Marra20  
/home/eeb-177-student/Desktop/eeb-177/CSB/unix/data/Pacific
```

note that -iname ignores the case in the file names

Challenge

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

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

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

```
>>
```

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

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 ownership (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:

open nano in your class-assignments folder and save the following file as dir.sh:

```
#!/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 added 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 `.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.