Shell Novice: The Linux Shell - Answers Version 1.1

Files and Directories

Relative path resolution

1. (4) `original pnas_final pnas_sub`

Reading comprehension

1. (2 or 3) `$ ls -r -F` or `ls -r -F /users/backup`

Default `cd` action

1. (3) It changes the working directory to the users home directory

Exploring more `ls` arguments

1. Print the allocated size of each file, in human-readable format

Creating Things

Renaming files

1. (2) `$ mv statstics.txt statistics.txt`

Moving and copying

1. (2) recombine

Listing directories and files

1. One command:

```
$ mv fuctose.dat sucrose.dat analyzed
```

Copy with multiple filenames

1. All of the specified files will be copied into the given directory
2. The following error will be generated:

`cp: target 'survey.txt' is not a directory`

The man page for `cp` explains why:

...

DESCRIPTION

Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

...

Listing recursively and by time

1. `$ ls -R -t` recursively lists the contents of the specified directory sorting by modification time, newest first

Pipes and Filters

What does `sort -n` do?

1. `-n` is short for `--numeric-sort` its used to compare according to string numerical value.

What does < mean?

1. Many commands accept input from standard input (stdin). By default stdin gets its input from the keyboard. In the first instance `<` is used to redirect stdin from a file instead of the keyboard, in the second `wc` is given a command-line parameter intructing it to recieve input from a file. The resultant stdout will often be different even if the same file is used.

Piping commands together

1. (1) `$ wc -l | sort -n | head -3`

Why does `uniq` only remove adjacent duplicates?

1. Attempting to match each line with every other would be very inefficient. Using `uniq` with `sort` is a common idiom.

There is an application on GitHub called ***`suniq`*** *which the author claims is a faster version of* ***`$ sort | uniq -c | sort -n[r]:`***

https://github.com/hyperair/suniq

Pipe reading comprehension

1. | Everything | the first five lines | the last three lines > reverse alphabetical order, so the contents of final.txt would be:

```
2012-11-05,raccoon
2012-11-06,rabbit
2012-11-06,deer
```

Pipe construction

1. `sort` & `uniq` e.g.

```
$ cut -d , -f 2 animals.txt | sort | uniq
```

Finding Things

Using Grep

1. (3) `$ grep -w of haiku.txt`

`find` Pipline reading comprehension

1. Find all files ending in .dat and output their line-counts in alphabetical order

Matching ose.dat but not temp

1. (3) `$ grep -v temp $(find /data -name '*ose.dat')`

Little women

1. To get tabulated data out (with totals) you would need somthing like the following for-loop:

```
$ for lw in Jo Amy; do echo ""$lw","$(grep -wo $lw littlewomen.txt
| wc -l)""; done
```

It's also worth mentioning the OR operator in Grep which enables each occurrence of each name to be listed with a simple one-liner:

```
$ grep -wo 'Jo\|Amy' littlewomen.txt
```

To make it slightly easier to distinguish the winner you could pipe to the `sort` command:

```
$ grep -wo 'Jo\|Amy' littlewomen.txt | sort
```

Transferring Files and Accessing a Remote Server

Exploring more `wget` arguments

1. Downloads each of the tar.gz files pointed at by the three URLs in the text file. Becuase they all have the same name the second file will get renamed .1 and the third .2.

Listing directories and files on a remote host

1. `$ ssh training1@10.20.208.208 ls -F /home`

Exploring more `scp` commands

1. `mkdir remote && scp training1@10.20.208.208:notes.txt remote`

Loops

Variables in loops

1. `$ for datafile in *.dat; do ls *.dat; done`

```
fructose.dat glucose.dat sucrose.dat
fructose.dat glucose.dat sucrose.dat
fructose.dat glucose.dat sucrose.dat
```

2. `$ for datafile in *.dat; do ls $datafile; done`

```
fructose.dat
glucose.dat
sucrose.dat
```

3. The first loop repeats an `ls` command, which uses a wildcard argument, three times. The second loop repeats an ls command, which uses the previously defined variable as its argument, three times. In both cases the loop is repeated three times because the first wildcard matches three files.

Saving to a file in a loop: Part 1

1. (1) Print fructose.dat, glucose.dat, sucrose.dat, and copies sucrose.dat to create xylose.dat

Saving to a file in a loop: Part 2

1. (3) All of the text from fructose.dat, glucose.dat and sucrose.dat would be contatenated and saved to a file called sugar.dat

Doing a dry run

1. Because the first loop does not "quote" the string we want to echo everything before the > symbol will be redirected to a file called analyzed-$file. The second one is the one we want to run.

Nested loops and command-line expressions

1. 4 6

Explain this loop

1. If frog11, prcb and redig were commands then each command would be executed with the argument `-limit 0.01 NENE01729B.txt...` but their not, so lots of errors will be displayed.

Shell Scripts

Variables in shell scripts

1. (4) An error because of the quotes around *.pdb

List unique species

1. Script:

```
#!/bin/bash
for file in "$@"; do
echo -e "\n"$file""
cut -d',' -f2 "$file" | sort | uniq
done
```

Find the longest file with a given extension

1. Script:

```
#!/bin/bash
wc -l $(find "$1" -name *."$2") | sort -n | tail -2 | head -1
```

Why record commands in the history before running them?

1. So you can:

- Double-check the commands you have just run
- Re-run commands without having to retype them
- Hand a P45 to the person who got trigger-happy with `rm -rf`

Script reading comprehension

Assuming the script was in the same directory as the three .dat files:

1. fructose.dat, glucose.dat and sucrose.dat would have their file names outputted to the screen.
2. fructose.dat, glucose.dat and sucrose.dat would have their contents outputted to the screen.
3. fructose.dat, glucose.dat and sucrose.dat would have their file name outputted to the screen, a .dat string would be added to the end of the list.