csce215 — UNIX/Linux Fundamentals
Fall 2021 — Lab Assignment 3

This assignment is intended to provide some practice and additional content for the material covered in lecture on Monday, September 13. You'll use pipes and redirection in a few different ways. The assignment is meant to be completed in the lab sessions on Wednesday, September 15, and must be submitted by 11:59pm on Friday, September 17. A total of 40 points are available.

1 Get started

As usual, this lab asks you to use a terminal window to execute some commands, and submit an `asciinema` recording of your terminal session. So the first step is to start that recording. You probably know how to do this by now, but if you've forgotten, you can refer back to the previous notes, or to Lab 1 or Lab 2. Remember that we can only award points for things that appear in the recording that you submit.

In addition, you'll be creating a few files for this assignment, so you should create a directory called `lab3` inside your `215` directory. Then complete the tasks below inside that `lab3` directory.

Use `asciinema` to record your terminal session. Create new directory called `lab3` to use for the tasks below.

2 Lend me your command lines

To get started, use `wget` to download this file, which contains the text of the classic play *Julius Caesar* by William Shakespeare.

```
https://cse.sc.edu/~jokane/215/shakespeare.txt
```

Use `wget` to download `shakespeare.txt`. *1 point*

This play is probably most famous for the eulogy of Julius Caesar, which begins:[1]

> Friends, Romans, countrymen, lend me your ears!

Suppose we wanted, for artistic reasons, to perform this eulogy in reverse. Let's use some pipes to get the script for our performance. We'll start small and add commands to the pipeline as we go. Be sure to execute commands that do each of the steps below.

1. Use `cat` to display the entire file.

   > Give a command that uses `cat` to display `shakespeare.txt`. *1 point*

2. We'll want to keep track of which line is which. So add to your command from the previous step a pipe that passes the output of `cat` into `nl`, which you'll recall from the notes is a command that numbers the lines of its input. Remember that `nl` is short for 'number lines', so it's the letter n followed by the letter l. If you've done this step correctly, you should see output that ends with a line like this:

```
1637        To part the glories of this happy day.              Exeunt.
```

   > Give a command that uses `cat` on `shakespeare.txt` with its output piped to `nl`. *2 points*

3. Wow, Shakespeare wrote some long plays! Let's use `grep` to search for the part we want. Adding to your command from the previous step, pipe the output of `cat` into `grep` to find the first line of the eulogy. We want to be lazy, so let's try to use a short search pattern, instead of typing the entire thing.

   For example, if we append

```
| grep countrymen
```

[1] You don't need to know anything about Shakespeare to complete this part of the assignment. It will be enough to know that you're looking for the 'Friend, Romans,. . . ' line, along with the next 12 lines after that.

to the previous command, we'll get 8 different matches. Try a few search patterns to come up with one that matches *exactly* the one line containing 'Friends, Romans, countrymen, lend me your ears!'.

Keep in mind that if your search pattern contains spaces or certain punctuation, you'll need to put it inside quotation marks. (We'll learn more about this next time.) For example, adding this

```
| grep Romans, countrymen
```

will not work, because `grep` will treat `countrymen` as the name of a file to search, instead of as part of the pattern. In contrast,

```
| grep "Romans, countrymen"
```

would be a valid search pattern, but would not be correct for this task because it matches two lines in the file instead of just the one we want.

Give a command that uses `cat` on `shakespeare.txt`, with its output piped to `nl`, and the output of `nl` piped to `grep` with a pattern that matches only the 'Friends, Romans, countrymen' line. *2 points*

4. Now that we've got the first line of the eulogy, let's use some other filters to get rid of the rest of the play. For this part, it will help to know that the eulogy we want to extract is 13 lines long. Remove the `grep` element from the previous command and replace it with an appropriate `head` command that eliminates everything *after* the eulogy. The output of the previous (which includes line numbers) can help. If you've done this correctly, you should have command consisting of `cat`, `nl`, and `head` that produces a long output ending with

```
896         And I must pause till it come back to me.
```

At this stage, you may (or may not) see a message saying

```
nl: write error: Broken pipe
```

If you do see this message, you can safely ignore it. (Explanation: Remember that all the programs in a pipeline run simultaneously. In this case, the `head` program may finish its job and terminate before `nl` has finished its work. If that happens, `nl` will complain that its standard output is not going anywhere.)

Give a command that uses `cat` on `shakespeare.txt`, with its output piped to `nl`, and the output of `nl` piped to `head` with appropriate options to eliminate everything after the end of Caesar's eulogy. *2 points*

5. Next, let's eliminate the part of the play *before* Caesar's eulogy. Add an appropriate `tail` command to the end of your pipeline to do this. If you've done it correctly, you should have 13 lines of output at this point.

Give a command that uses `cat` on `shakespeare.txt`, with its output piped to `nl`, the output of `nl` piped to `head` with appropriate options to eliminate everything after the end of Caesar's eulogy, and the output of `head` piped to `tail` with appropriate options to eliminate everything before the start of Caesar's eulogy. *3 points*

6. Almost there! We need to reverse the order of the lines. Add one more command to the pipeline to do this. (Hint: This is easy. Check your notes if you're not sure how to do it.)

Give a command that uses `cat` on `shakespeare.txt`, with its output piped to `nl`, the output of `nl` piped to `head` with appropriate options to eliminate everything after the end of Caesar's eulogy, the output of `head` piped to `tail` with appropriate options to eliminate everything before the start of Caesar's eulogy, and the output of `tail` piped to a command that reverses the order of the lines. *3 points*

7. Finally, since we want to store the results of this work for later, add an output redirection at the end of your pipeline to save the results in a file called `eulogy.txt`. Use `cat` to verify that this file correctly contains the 13 lines of Caesar's eulogy in reverse order.

Give a command that uses `cat` on `shakespeare.txt`, with its output piped to `nl`, the output of `nl` piped to `head` with appropriate options to eliminate everything after the end of Caesar's eulogy, the output of `head` piped to `tail` with appropriate options to eliminate everything before the start of Caesar's eulogy, the output of `tail` piped to a command that reverses the order of the lines, with the output of the entire pipeline redirected to a file called `eulogy.txt`. Use `cat` on `eulogy.txt` to ensure that your redirection worked correctly. *2 points*

3 On the halfshell

The next two sections will utilize the file at this URL:

```
https://cse.sc.edu/~jokane/215/tmnt.txt
```

Use `wget` to download it. Then take a look.

Use `wget` to download `tmnt.txt`. *1 point*

As you can see, this file contains the records of an epic three-hour battle between the Teenage Mutant Ninja Turtles and the forces of evil. Each line records the time and details about when one of the ninja turtles defeated one of the bad guys.[2]

Though the log you downloaded covers the period between 9am and noon, the most important parts occurred between 9am and 10:59pm. Your job is to *create a pipeline that outputs the lines from this interesting part, i.e. those that happened strictly before 11:00am, ordered from earliest to latest.*

This time, rather than walking through things step-by-step, we'll leave you you to figure out an appropriate series of filters on your own. Don't hesitate to consult your notes and possibly the `man` pages to solve this problem.

As a hint, notice that there are two separate things that need to happen: (1) Eliminating events whose time is outside the important 9:00–10:59 time window, and (2) ordering the

[2]Yes, this is fictional and maybe a little silly. But the ideas here are the same as, for example, looking for patterns of errors in real system logs.

events correctly. You can tackle these steps in either order, but you should produce one pipeline that does both.

Design and execute a pipeline that reads input from `tmnt.txt` and outputs only the lines with times before 11am, ordered from earliest to latest. *10 points*

Now let's count the number of bad guys defeated in that time. Add a filter to the end of your answer for the previous part to count the number of lines. Use appropriate options to ensure that *only* the number of lines appears in the output, with no other numbers. When you've got both this part and the previous part correct, your pipeline should output a single number, `650`.

Add to your previous pipeline an additional step that outputs the number of bad guys defeated by the turtles before 11am. *3 points*

4 Which turtle is best?

Our last task will be to process the data in `tmnt.txt` to answer a different question:

> Which ninja turtle is the best ninja turtle?

For our purposes, we'll say that the turtle that defeats the most bad guys is the best turtle. So we need a pipeline that outputs 4 lines, one for each turtle (`Leo`, `Raph`, `Mikey`, and `Donatello`). Each line should contain the turtle's name, preceded by the number of bad guys that turtle vanquished. These four lines should be in order from most to least.

Here's an example of what the output should look like, but with sample numbers instead of the real answers:

```
28 Leo
15 Mikey
13 Raph
11 Donatello
```

You should create a pipeline that produces a table like this one, based on the data in `tmnt.txt`. This will take several steps, but all of them can be done with filters that we've talked about.

- Use one filter to extract, for each line, the second word on that line. This is the name of the turtle. After this step, each line should have just the name of one of the turtles.
- Use another filter to rearrange the lines so that identical lines appear next to each other.
- Use a third filter to count the number of copies in each set of identical lines.
- Use a fourth filter to put these remaining lines in decreasing numerical order.

All four of these should be combined into a single pipeline.

Design and execute a pipeline that reads input from `tmnt.txt` and outputs a table in the format above, showing which turtle defeated the most bad guys.

(If we don't see the fully correct answer, we'll try to award partial credit by looking for correct solutions to the intermediate steps suggested above.) *10 points*

5 It's over!

Just as in Lab 1 and Lab 2, you should end the recording with `Ctrl-D` or `exit`.[3] Optionally, replay the recording to ensure that it shows you completing each of the tasks listed in the checklist boxes above. Then upload the completed recording to the Dropbox site and be sure to submit the assignment.

[3]Quick aside: Now we know why `Ctrl-D` ends the asciinema recording! The shell that's being recorded is reading from its standard input. When we send it a `Ctrl-D`, that signals that the standard input has ended, so that shell terminates.