Basic - Unix
Unix and the Shell

EECS 201 Winter 2022

Submission Instructions

Answer the bolded text that begins with a "**Q**". This assignment is an "online assignment" on Gradescope, where you fill out your answers directly.

Preface

This assignment should be fine on Linux and macOS. That being said, if you run into issues on macOS, like a command completely not running as expected, try using the course server or an Ubuntu 20.04 VM.

1 Redirection

Recall from lecture that we can *redirect* the inputs and outputs of a command. `echo` is a command that will print out a string (taking into account any expansions you embed in it). For example, I could use `echo "Welcome $HOME"` to print out `Welcome /home/brandon`. Try modifying this command so that it saves that output to a file.

For the sake of formatting for the Gradescope automated grader (feel free to do whatever outside this assignment):

- Handle the redirections in the order described in the question
- Have only a single space before whatever redirection operator you choose
- Omit space between the operator and its file (e.g. `<file`, not `< file`)
- Only use an explicit file descriptor number if it needed (e.g. `>file`, not `1>file`)

Q: Write a command that saves the output of `ps` to a file named `processes`.

Q: Write a command that appends the output of `w` to a file named `user-log`.

Suppose I had an application `app` that *reads data from standard input*, and produces its *output on standard output* and reports *error messages on standard error*. Let's say we have a file called `input.dat` in the current directory. Assume that `app` is in the `PATH` so you can invoke `app` just as `app` (no `./`).

Q: Write a command that feeds `app` data from `input.dat`, saves the output to `output.dat`, and saves the error messages to `log.txt`.

2 Pipeline

Recall from lecture that commands can be chained together to form a pipeline, where one process passes its output to the input of the next. For example, I could create this pipeline `cat file1 file2 file3 | rev` to concatenate three files and output the result with `cat` and then reverse the output of `cat` with `rev`.

For the sake of formatting for the Gradescope automated grader (feel free to do whatever outside this assignment):

- Put one space around pipes (e.g. `cmd1 | cmd2`, not `cmd1|  cmd2`
- Put only one space around arguments (e.g. `cmd1 arg1 arg2`, not `cmd1 arg1   arg2`, i.e. as the question presents the commands)

- Strip headers from the main piece of data before further processing (e.g. remove the header before isolating column 8)
- Use all of the given commands.

Q: Using only the commands below, create a pipeline that creates a list of storage devices and info sorted by space usage. There should not be any headers in the resulting output. As you do this question, try testing out your pipeline by running it.

- `tail -n +2`: This will print the input from the second line onwards (`tail` prints the ends of file; with these arguments we can specify where to start printing from)
- `sort -h -k5`: This will sort by the 5th column using a human numerical sort
- `df -h`: This will print out disk usage of various storage devices in a more human readable fashion. The 5th column is the percentage used. The first line of output is the header line, providing labels for each column. **You probably wouldn't care about this header line**.

Q: Using only the commands below, create a pipeline that creates a sorted list of the user's currently running *commands* with no repeats. As you do this question, try testing out your pipeline by running it.

- `awk '{print $11}'`: This will print out the 11th column of each line of input
- `ps ux`: This will list out all of the user's processes. The 11th column is the command column. The first line of output is the header line, providing labels for each column. **You probably wouldn't care about this header line**.
- `sort`: This will sort inputted lines and output them in a sorted order
- `tail -n +2`: This will print the input from the second line onwards (`tail` prints the ends of file; with these arguments we can specify where to start printing from)
- `uniq`: This will take inputted lines and omit repeated, adjacent lines: for example if "hello world" occurs on three **adjacent** lines, only one "hello world" is outputted.

3 Files

Recall from lecture that files have metadata that correspond to what permissions various users have for interacting with them: read, write, execute. These bits are grouped together to form a 3-bit octal (base 8) digit, in the order of `rwx` (read, write, execute). If you are unfamiliar with binary (base 2) representation, it's like decimal except each place is a power of 2, and the only option for each place is 0 or 1. For example, `110` is

$$1 * 2^2 + 1 * 2^1 + 0 * 2^0 = 4 + 2 + 0 = 6$$

.
If we look at what that means for permission bits, `110` maps to `rwx` (read, write, execute), so that means that an octal `6` means "readable and writable".
These files also have metadata that track ownership: what user owns the file and what group owns the file. Together, a file has three sets of these permission bits, one set of three corresponding to permissions for the user owner, one set for the group owner, and one set for everyone else not the user and not in the group ("other").

Q: Suppose you have a file that you wish to make readable, writable, and executable by the user who owns it, and only readable and executable by everyone else? What three digits should you provide to `chmod`?
Your answer must contain only three digits with no spaces.

Q: Suppose you have a file that you wish to make readable, writable, and executable by the user who owns it, but only readable by the group that owns it, and completely inaccessibile to everyone else. What three digits should you provide to `chmod`?
Your answer must contain only three digits with no spaces.