CS 180 Notes – 1/27/16

Pre-Class Notes

- Make sure you document how long each assignment took and what parts of the assignment were particularly troublesome.

Bash-Shell Filters

- When we do something like *prog A | prog B* or *cat foo.txt | prog B*, a few things are going on behind the scenes. (" | " is called a pipe)
    - We take the stdout from *prog A* and pipe it into stdin for *prog B*.
    - *prog B* has absolutely no idea this is happening. Bash is handling this issue.
    - It is similar to what we saw in CS120 with *prog B < foo.txt*.

Examples of filters in use

- It is not uncommon for computers to have multiple ip addresses. (To check this, use *ifconfig* in UNIX and *ipconfig* in Windows.)
- There are also different classes of IP addresses
    - A class C address is of the form N.N.N.H, where 'N' is network octets and 'H' is host octets.
    - Similarly, class B addresses is of the form N.N.H.H.
    - Class A addresses are of the form N.H.H.H.
- What if we wanted to know how many class B addresses were on our computer?
    - *cat ip.txt | grep "IP Address" | cut –c 45- | cut –d –f1,2 | sort | uniq | wc –l*
        - Display "ip.txt" (assuming you did *ifconfig > ip.txt*)
        - Search for "IP Address" using *grep*
        - Only get the actual IP addresses using *cut*, from character 45 onwards. (Trial and error to figure the number of characters)
        - *cut* again using a period "." as a delimiter. Only extrac the first two fields.
        - *sort* the addresses
        - *uniq* removes duplicates of our IP addresses
        - Finally, *wc –l* gives us the final number of unique addresses, which is what we were after.
        - Final output: 2

- Another example using decl.txt (The Declaration of Independence, arguably the greatest American document of all time). What if we wanted to find out how many unique words were in the document?
    - *cat decl.txt | fmt –w1 | tr –d [:punct:] | grep –v "^$" | sort | uniq –i | wc –l*
        - Display decl.txt
        - Format display so that there is one word per line
        - Translate (delete with -d) all punctuation
        - **Inverse** search for empty lines (in order words: don't show them)

        - Sort the remaining words
        - Remove duplicates (with case insensitivity)
        - Count the remaining words

Quick Examples

- All users on a system: *cat /etc/passwd | cut –d: -f1 | wc –l*
- Path on each line: *echo $PATH | tr ':' '\n'*
- Justify left and right with padding: *cat decl.txt | par 80j*

Writing our own filter (mynl.c)

- Included in this .zip file is sample code written to display line numbers to a file

Homework #2 Comments

- Add Doxygen comments to main and all helper functions you write!
- Must remember to connect to stdin. Students traditionally lose points here.
- You may have to pass over the command line twice to get filenames
    - First time – get the number of files
    - Second time – Dynamically allocate space for each file
    - Make a flag to handle this. Do not create two types of command line passing functions.
- Number all lines in *line_numbers*, including blank lines
- Suggested order is included in handout
- All output should go to stdout
- **Read a character, write a character!** Don't try and write lines or use a buffer
- Handle errors with useful messages
- You may assume all arguments are valid.