Eric Paulz
CPSC 3220-001

Project 1

1. Please provide command-line code and full-sentence English interpretation of the results for the following:
    a. Identify 4 image file formats and 4 source code formats present within fs1.zip and fs2.zip.
        i. *fs1:* **grep -E -i '.*\.jpg$' tape* | wc -l**
            1. *Explanation*
                a. **grep -E -i '.*\.jpg$' tape*** attempts to match any file ending with the extension .jpg or .JPG within all of the tapes in fs1. The -E flag lets me extend the regular expression to include many different patterns instead of just one. The -i flag tells grep to ignore case. This is important because in Linux .jpg and .JPG are the same thing, so we wouldn't want grep to count them separately.
                b. **wc -l** prints the total number of lines of output, or how many files were found with a matching extension.
            2. I continued to run this same command in a trial-and-error fashion with different known file extensions. I identified the following extensions:
                a. Image: .jpg, .ppm, .gif, .tiff
                b. Source: .c, .cpp, .html, .java
        ii. *fs2*: same command as for fs1 except instead of tape* being grep's target I changed it to ls-redaction.txt.
            1. Image: .jpg, .ppm, .gif, .tiff
            2. Source: .c, .cpp, .html, .java

    b. How many files appear cumulatively? How many were from 1990-1995? How many from pre-1990?
        i. *fs1*: There are 2666 .jpg, 72 .ppm, 13722 .gif, 1646 .tiff, 15214 .c, 168 .cpp, 3456 .html, and 353 .java. This gives us a total of 37297 files. To identify what dates the files were from, I piped the previous grep command to an awk command before printing the word count. The command looks like this:

```
grep -E -i '.*\.jpg$|.*\.ppm$|.*\.gif$|.*\.tiff$|.*\.c$|.*\.cpp$|.*\.html$|.*\.java$' tape* | awk '$7 >= 1990 && $7 <= 1995' | wc -l
```

        The awk portion of this command grabs the 7th column of output, which in this case is the year, and only returns those rows where the year is within the specified range. Running this command tells me that 36928 of the files are from 1990-1995, and only 368 of them are from before 1990.
        ii. *fs2*: There are 578229 .jpg, 95 .ppm, 66201 .gif, 46380 .tiff, 6605 .c, 2515 .cpp, 326286 .html, and 6045 .java. This gives us a total of 1032356 files. Using the same command as I did in fs1 to identify dates (except this time the year was in column 8 instead of column 7), I found that only 7 of the files are from 1990-1995, and 228243 of them are from before 1990.

    c. What fraction of files have no registered file extension? What interpretations do you have of this?
        i. *fs1:*
            1. **cat tape* | wc -l**
                a. Returns the total count of files in all tapes.
                b. Output is 395747
            2. **grep -E '.*\..*$' tape* | wc -l**
                a. Returns total count of files ending in *.something* in all tapes.
                b. Output is 219276

Comparing these two outputs, we see that $\frac{395747-219276}{395747} = \frac{176471}{395747} \approx 0.45$.
So, about 45% of the files in fs1 have no registered file extension.

ii. *fs2:*

1. **cat ls-redaction.txt | wc -l**
    a. Returns the total count of files in ls-redaction.txt.
    b. Output is 5760770
2. **grep -E '.*\..*$' ls-redaction.txt | wc -l**
    a. Returns total count of files ending in .*something* in ls-redaction.txt.
    b. Output is 3752562

Comparing these two outputs, we see that $\frac{5760770-3752562}{5760770} = \frac{2008208}{5760070} \approx 0.35$.
So, about 35% of the files in fs2 have no registered file extension.

iii. *Interpretation:* An extension-less file could mean a few different things, though it hard to tell exactly since we can only see xxx for the filenames. I assume they are some kind of file (like .txt) that is saved without an extension. For example, the tape files in fs1 don't have an extension in their name. Another possibility is that some of them are empty directories, so the directory name is the last thing on the line.

2. Questions re SWOT on AFS /dev mappings for lights and dams.
    a. It is technically possible to use AFS and /dev-like approaches to map
        i. a) Every fixtured light on Clemson campus to an Internet-accessible filesystem?
        ii. b) Every dam in SC to an Internet-accessible filesystem?
    b. For each, argue SWOT.
        i. a) Lights
            1. *Strengths:* Ability to customize lighting schedules via an app. Smart lights learn when certain rooms are and aren't in use, increasing efficiency and reducing costs. Gives facilities managers the ability to monitor and control lighting throughout the entire building.
            2. *Weaknesses:* Potential security threats. Requires greater initial investment and more specialized installation. Would be difficult to replace a buildings existing lighting system with a connected system… better suited for brand new buildings.
            3. *Opportunities:* If all new buildings adopt a connected lighting system, overall power efficiency will increase, which will either reduce power station resource usage or simply free up that power to be used elsewhere. Could potentially integrate LiFi technology, so the connected lighting fixtures also produce Internet access to a room's occupants.
            4. *Threats:* If an entire system's lighting network can be controlled by one central software console, then a breach of that console could lead to security issues. Similarly, if employees can control their office's lights with a mobile app, that is just another vulnerability. If a bad actor can gain access to the building's lighting, does that mean he/she can gain access to the entire network?
        ii. b) Dams
            1. *Strengths:* Real-time monitoring of the entire dam network. Fast, distributed responses to emergencies and failures. If a dam upstream fails, any dams downstream can be notified immediately to take precautions to prevent domino failures.
            2. *Weaknesses:* Would be very expensive. Thousands or millions of IoT devices would be required. Security issues. Many existing dams are very old and not set up to integrate modern technologies like IoT.

3. *Opportunities:* Could save many lives and dollars if implemented correctly. IoT devices could automatically send warnings to the public if a dam failure is possible/imminent. Would give us the ability to control dams remotely; speed and convenience.
4. *Threats:* Could cost many lives and dollars if control of the system fell into the wrong hands. Human error is a factor… a small mistake could lead to a distributed catastrophe.