# CS367 Lab 3

Begin by copying the file **/usr/local/cs367/lab03.tar.gz (**on **cs367.sou.edu)** into your home directory. Untar and uncompress the file and then **cd** into the directory **lab03** before answering the questions below. Submit a hard-copy of your answers. Do not make any assumptions about the contents of files, directories, current variable values, etc.; your commands should be generic enough to work in any situation (as applicable). Use only commands/flags/etc. discussed in class.

**Part A.** For each question, write a *single* **bash** command line (possibly consisting of multiple filters in a pipeline). If your command line generates output, your command should display ***only*** the fields and lines requested in the question.

1. Each line in the file **log** is a record of web file accesses, consisting of the fields, in order: accessing IP address, web file accessed, and number of accesses. Output the contents of **log** sorted by number of accesses in descending order (i.e., greatest number of accesses first).
2. Output all of the web files listed in the file **log** accessed by IP address **140.211.106.11** sorted in ascending sequence by number of accesses.
3. Use **ls** and a pipeline (***not*** pathname expansion) to output the names of all files in the current directory (**lab03**) that begin with an '**l**' (a lower case '**L**') followed by any three characters, followed by (and ending with) "**.txt**", sorted in ascending sequence by size.

The rest of the problems in this Part (and Part **B**) all use the file **cds.db**. This is a text file containing a partial catalog of my music CD collection. Each line is a record for a single CD, consisting of the fields, in order: artist name, CD name, record label, and year of release. The fields are separated by colons (**:**). All questions below refer only to CDs in this file.

4. List in alphabetical order the record labels which **Van Morrison ("Morrison, Van")** has released CDs on. Do not display duplicates.
5. List the artist and album name (in that order) for all CDs, sorted in chronological order (oldest first). Label each field in the output (e.g., "ARTIST: Mitchell, Joni  ALBUM: Hits"). Replace all instances of "**Morrison, Van**" with "**Van the Man**". Redirect the output to a file named **albums.txt**.

**Part B.** In this problem you will write a series of commands that create an html table of cds, sorted by year of release, that could be put on the web. An example of the resulting html file is linked from the class website. You will need the files **cds.db, cdstop.txt,** and **cdsbottom.txt** (all in the **lab03** directory).

Your commands should do the following:

**1.** Make a copy of **cdstop.txt** called **cds.html**. **cdstop.txt** contains the start of an html file defining the appropriate table.

**2.** Write a single pipeline command that generates an html table row entry for each of the cds in **cds.db** and appends these to **cds.html**. Each row entry will contain the album title, artist, and year of release, in that order. Use an **awk** program file for any **awk** portion of this pipeline (that is, put your **awk** program in a text file and execute it using the appropriate flag). ***The table rows should be sorted by year of release, from oldest to newest.*** Each row entry should be of the form (using the first cd as an example):

```
<tr>
<td align=left> Astral Weeks </td>
<td align=left> Morrison, Van </td>
<td align=right> 1968 </td>
</tr>
```

**3.** Finally, append the contents of **cdsbottom.txt** to **cds.html**. **cdsbottom.txt** contains the end of the table and html file.

**4.** Create a shell script that contains the three command-lines you used for steps **1-3** above. You should be able to execute this shell script whenever the cd collection changes to create an updated web page. What command-line would you type to execute your shell script?

Submit the following for Part **B**:

**a.** the awk program file used in step (**2**)
**b.** the shell script you created in (**4**).
**c.** the answer to the question posed in (**4**).

*Note:* You can see what your final **cds.html** file looks like by copying it (using **scp/WinSCP**) to your PC/Mac and then using "**open file**" in your web browser, or by scp'ing the file to your **public_html** directory on **webpages.sou.edu** and then accessing the url **http://webpages.sou.edu/~*yourusername*/cds.html** in your web browser.

**Part C.** For this part you will create a **~/.bash_profile** shell initialization script. Your shell script should contain the following command-lines, in the order indicated (do not worry about numbering your answers for this part).

1. Set your prompt to a string of your choice. Your prompt must contain your working directory (using the appropriate bash prompt mechanism).

2. Add your current directory to your command search path as discussed in class.

3. Create an alias that performs a long format recursive listing (i.e., listing all sub-directories recursively) of the root directory piped to more.

4. Create a variable that contains your first name.

5. Create another variable that contains your last name.

6. Create an alias that outputs your first and last name using the variables defined above (i.e., this alias should still function if you change the values of the variables).

**Part D.** Logout and back into the system so that your Part **C .bash_profile** is executed and then cd into the **lab03** directory. For each question, write a *single* **bash** command line (possibly consisting of multiple filters in a pipeline).

1. Delete the alias you created in question **C.3**.

2. Change the value of the variable you created in question **C.4** so that it contains just your first initial and a period (i.e., my variable value would change from **"Kevin"** to **"K."**).

3. Execute the alias you created in question **C.6**.

4. List the names of all files in the current directory that contain the word "**Sahr**". Do not display duplicates.

5. Set a variable equal to the output of the command described in question **10** (remember that this must work without making assumptions about what that output was).

6. Concatenate the contents of the files specified in the variable you created in question **5** into a single file named **"them.txt"**.

7. Output "**No Sahr in *number* lines**", where ***number*** is replaced with the number of lines in **them.txt** that do **not** contain the word "**Sahr**". Do not use a pipeline.

**Part E.** Write a bash shell script that uses a for-loop to do the following for each file in the current directory whose name ends with "**.txt**":

**1.** output "**** Processing file *fileName***", where ***fileName*** is the name of the file.
**2.** output all lines in the file that contain "**Sahr**", replacing all instances of "**Sahr**" with "**SAHR**".

Submit a hardcopy of your shell script. Note that the **lab03** directory is a good directory in which to test this shell script.