

# LAB 3: WEB & UNIX SUPPLEMENT

## Section 1

1. Make a new webpage for your “Computer\_class” items, linked to your main webpage. To do this, make a new index.html file in your Web/Computer\_class folder. Edit this webpage to look something like the example page I created:

[http://www.geo.utep.edu/pub/bkonter/geol\\_5215/Computer\\_class/](http://www.geo.utep.edu/pub/bkonter/geol_5215/Computer_class/)

---

## Section 2

1. Open a UNIX window on your computer using either X11 (Macs) or Cygwin (PCs, issue the startxwin.bat command).

2. In the terminal window, type

**pwd**

*What was the response to this command? Where are you currently residing in the computer directory structure?*

3. Make sure that the //geobase/GEO5215 server is mounted on your computer, and move to your user folder in that directory. *What is the command for doing this?*

4. Change directories to your Lab/Lab2 directory that you made in the Lab2 exercise assigned last week. Verify that you are in subdirectory **Lab2**. *What command did you use to do this?*

5. Now copy the file **employees.dat** from the course website and save it to your GEO5215 user folder:

[http://www.geo.utep.edu/pub/bkonter/geol\\_5215/temp](http://www.geo.utep.edu/pub/bkonter/geol_5215/temp)

6. Next copy the employees.dat file from your user folder to the **Lab2** directory. Since you should still be inside your Lab2 directory, an example of how to do this is:

**PCs:** `cp //geobase/GEO5215/username/employees.dat .`

**Macs:** `cp /Volumes/GEO5215/username/employees.dat .`

Remember, the “.” means “put it here, where I am executing the command”. To check on your file, type:

**ls**

7. The file **employees.dat** consists of information on employees. The column titles listed below are only for your information and do not appear in the file. The file follows:

<b>Dept</b>	<b>Last Name First Name</b>	<b>Date Hired</b>	<b>Salary</b>
mgt	Cooper John	06151995	66000
mgt	Davidson Darla	04151992	69500
mgt	MacDonald George	06151985	70000
act	Smith Thomas	04102002	56000
act	Smith Alecia	04121991	65000
mis	MacLeod Janice	01021977	90000
mis	Mack Joe	02252003	85000
mis	Winslow Sarah	02151995	58000
adm	Smith Dexter	01021975	100000
mis	Bennett Joan	08152001	79000
mgt	Neason Elizabeth	10251998	65500
act	NeSmith Donald	11301966	99500

To see what the file looks like, you can type:

**cat employees.dat**

### **Sort Command**

**Sort** sorts the file on a line-by-line basis. If the first characters on two lines are the same, sort looks at the second characters to determine the proper order. This process continues until sort finds a character that differs between the lines.

If lines are identical, it does not matter which one sort puts first. It also uses machine collating sequence--which in this case is ASCII code. Some important features of ASCII code is that capital letters have higher priority than lower case letters. Numbers have a higher priority than letters. An important point to remember is that **sort** is a filter and does not change the contents of the input file. It takes the contents of the specified input and outputs it in a sorted fashion. In other words, it filters its input. If the output is not redirected to a file, the output goes to **stdout**--which means the terminal screen. The sorted output will not be saved unless the output is redirected to a file.

8. First, type the following command:

```
sort employees.dat
```

*What is the order that **employees.dat** is sorted in?*

9. The first sort will sort the **Dept** field in alphabetic order. The sort command will sort the first field on the machine's collating sequence if no options are specified. Listed below are some options that you will use to control the way in which the **sort** command works. There are other options available that are explained in the **man** page for **sort**.

**-b ignore leading blanks** - Blanks (TAB and SPACE characters) are normally field delimiters in the input file. Unless you use this option, sort also considers leading blanks to be part of the field they precede.

**-f fold lowercase into uppercase** - This considers all lowercase letters to be uppercase letters. Use this option when you are sorting a file that contains both uppercase and lowercase text.

**-n numeric sort** - When you use this option, minus signs and decimal points take on their arithmetic meaning and the **-b** option is implied. The sort utility does not order lines or order sort fields in the machine collating sequence but in arithmetic order.

**-r reverse** - Reverses the order of the sort. If it is a numerical sort, the output will be in descending order.

You can sort on the different line fields. There are five line fields in **employees.dat** (dept, last, first, date hired, and salary). These sequences are

bounded by blanks or by a blank and the beginning or end of a line. You can use these line fields to define a sort field. You can instruct sort to skip several fields if you wish. If you wish to sort on the **first name** field, you would use **sort -k3 employees.dat**.

First, sort on the field for last name:

```
sort -k2 employees.dat
```

Look at the sorted file. *Are all the names sorted in alphabetical order? Give a brief description of how the names are sorted.*

10. There is a problem because MacLeod comes before Mack. The sort command put "L" before "k" because it arranges lines in the order of ASCII character codes, or capitol before lower case. Also NeSmith comes before Neason.

In this ordering, uppercase letters come before lowercase letters. You can use the **-f** option to have the sort command ignore this and sort alphabetically. This option will fold the lowercase letters into uppercase letters and correct the problem from the sort above. The option is placed before the field number that is to be used in the sort. Sort the file again using the following command:

```
sort -f -k2 employees.dat
```

*What happens when you sorted it this time?*

11. The next sort will be on the **"date hired"** field. Also save these next three sort routines into files by using the **redirect ">"** symbol. Remember that the redirect symbol saves, to a file, output that would normally go to the screen (**stdout**). Type:

```
sort -k4 employees.dat > hired1.dat
```

Use the cat command to list out the file **hired1.dat** to see the results. *Are the hire dates sorted in order? If not, what has happened?*

12. This sort **did not put the numbers in order** but put the shortest name first in the sorted list and longest name last. It turns out, sort picked up a few invisible spaces (or blanks) after the third line field, and decided that these were columns. T

One solution to this problem is to eliminate the leading blanks by using the **-b** option. However, since **dates hired** is a numeric field, you can use the **-n** option.

Type:

```
sort -n -k4 employees.dat > hired2.dat
```

*What is the result of the sort?*

13. When you sorted on the **date hired**, it basically sorted on the **month** hired (the first two digits). The numeric sort treats this date field as a single number and sorts in a true numeric order. Hence, the months 01 (Jan) will come before the months 02 (Feb) regardless of the day and year of hire. You may want to sort on the **year** hired instead. This can be done also. You can not only skip line fields but you can also skip characters in one line field as well. The **-k4.5** jumps to the fourth line field and then jumps to five characters before it sorts. Remember you must take care of the blank spaces also. Type:

```
sort -nb -k4.5 employees.dat > hired3.dat
```

*What was the result?*

#### 14. Sorting on more than one field

UNIX also allows you to sort on more than one field. One example would be to

sort on the department and on salary. We can see if people in one department are getting paid more than other departments. Of course there are other things that may be factors on salary such as years on the job. Type:

```
sort -k1 -k5n employees.dat
```

*What was the result? Were both columns sorted?*

15. As you can see, there are complications! First of all, note that the **n** was used after the **5**. That is because the first field is alphabetic and we do not want to sort the entire file with a numeric sort. So in this case, the **n** is used after the field number.

Next, the command line instructs sort to sort on the entire line (-k1) and then make a second pass, sorting on each entire line. Look at the first field and second field in the first two lines. The sort routine sorts on the first field and then goes to the second field. It matches on the names and does not go further. In order to stop the sort routine from going past the -k1 field, you need to define where the first sort ends--in this case, it will be **-k1**. This will stop the sort before going to the next field and will then go on to the salary field. Type in the next command.

```
sort -k1,1 -k5n employees.dat
```

*What were the results of this output?*