Handout Review Linux Search, Extract and Archiving Files

1. Launch Ubuntu: Make sure that you're in YOUR **home/user** Directory. Enter **pwd (**present working directory) command to verify location.
    a. Let's review the **make -p** command (used to create parent and subdirectories), **rmdir** command (used to remove a directory; requires an empty directory to work properly), and **rm -r** command (removing directories and subdirectories no matter what files or information exist in any of the directories or subdirectories).
        i. Use the make -p or –parent command to create the following directory structure: ***FirstDir/SecondDir***
            a. Use the following command string: **mkdir -p FirstDir/SecondDir**
            b. Note: FirstDir is the parent of SecondDir (or subdirectory of First Level)
            c. Enter **ls -R** command to verify the directory structure resides in YOUR home directory.
        ii. Change directory to First Level; use command **cd FirstDir**.
            a. Create a file named **NewFile1.txt** using the touch command
            b. Enter **touch NewFile1.txt**
            c. Enter **ls -l** to verify the file resides in this directory. The file NewFile1.txt and directory SecondDir should appear in the directory information list.
            d. Use relative reference to access the SecondDir directory
            e. Enter the following command **cd ~/FirstDir/SecondDir**
            f. Note the (tilde) ~ means HOME then reference the attached path information to direct location to the SecondDir subdirectory. Since we were already in the FirstDir directory, we could have simply entered cd SecondDir to get to the directory. Remember practice, practice, practice
        iii. The path information should indicate current location path is SecondDir. Enter pwd to verity location as well. Should return **/home/user/FirstDir/SecondDir**
            a. Create a new file in the SecondDir directory using the touch command. Name it NewFile2.txt. Use the following command **touch NewFile2.txt**
            b. Use **ls -l** command to verify that the file Newfile2.txt resides in the SecondDir directory
            c. Go back to FirstDir. Several ways, the easiest is to Enter command **cd ..** (or cd space dot dot,; this command always takes is back up one directory level.) Or you could use the relative reference command **cd ~/FirstDir**.
            d. Attempt to remove the directory SecondDir by using the **rmdir SecondDir** command.
            e. What was the response?
            f. Use the following command to access the SecondDir. **cd SecondDir**
            g. Enter **ls -l** to see what information is in the file. Enter the **rm NewFile2.txt** to delete the file.
            h. Go back to the FirstDir level…remember **cd ..** command is the simplest way to move up one directory. Don't' forget to use up and down arrows to access all history command information.
            i. Repeat **step iii d** above using the **rmdir SecondDir** to delete the directory
            j. Go to home cd ~
            k. Use the rm -r to remove the FirstDir and all content (whether empty or not). Recall the file NewFile1.txt resides in this directory. Enter **rm -r FirstDir**.

2. Create Hard and Symbolic Links. Hard links allows the data of a file to have more than one name in separate places in the same file system. Such a file with more than one name for the same data is called a hard-linked file. Refer to the illustration in Figure 1. There are three files (foo, bar, zoo) that share the same location(s) in the data file structure (low level). The inode is simply a record in a disk table. Remember, when a file is stored on a disk, it can be stored in different locations based on sized. The inode is the index used to reference the stored information. It is unique for each file and directory.

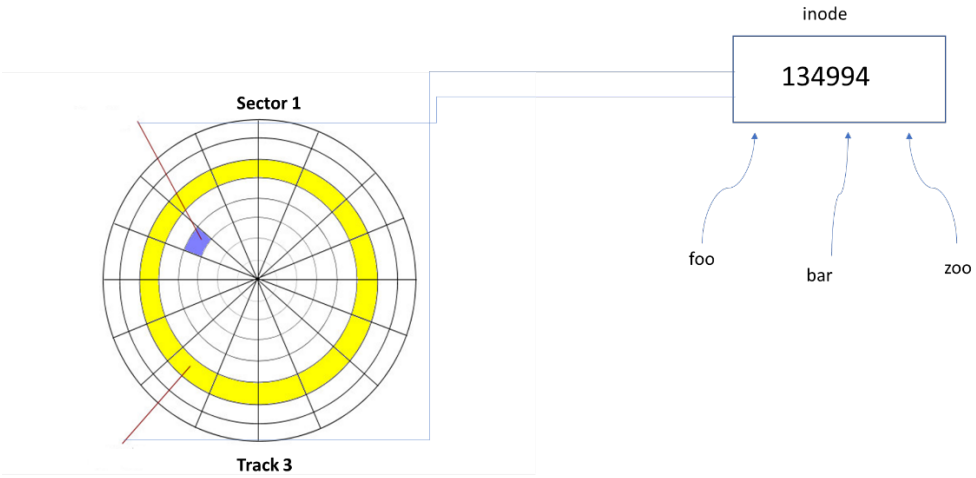
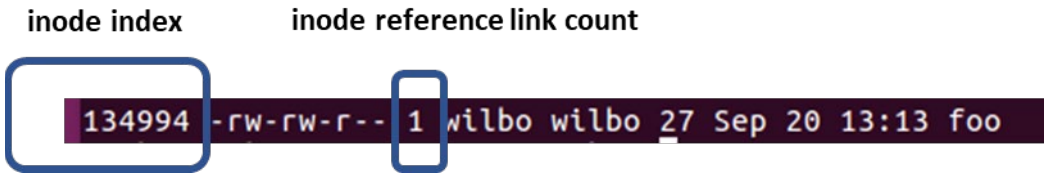
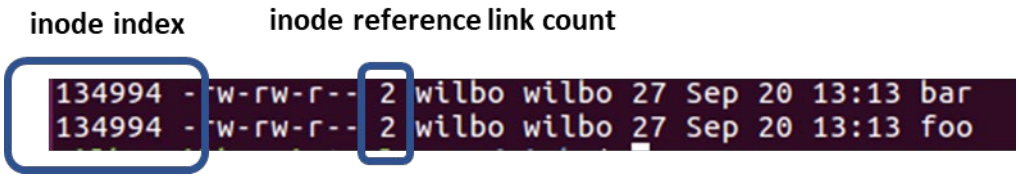


Figure 1: inode Index

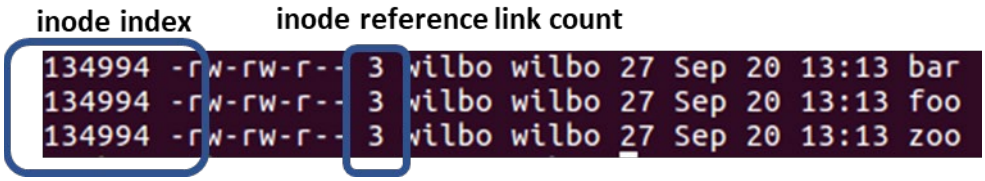
a. Go to your HOME Directory, **cd ~**
b. Create a subdirectory named Links. **mkdir Links**
c. Change directory to Links, **cd Links** or **cd ~/home/user/Links**
d. Create a text file named foo, Enter **touch foo**
e. Write the following text into the file using the echo command
   i. **echo "Linux is a very powerful operating system" > foo**
   ii. Use the **cat foo** command to verify content of foo
   iii. Enter command **ls -li** to show long list information with **inode index** number.



   iv. Use the following command to create hard link between foo and bar
   v. ln foo bar
   vi. Enter **ls -li** command
   vii. Note that foo and bar share the same inode index 134994. Deleting or removing either of the files means one will still reference the disk storage location(s).



viii. Hard link bar to zoo, use command **ln bar zoo**, then **ls -li**

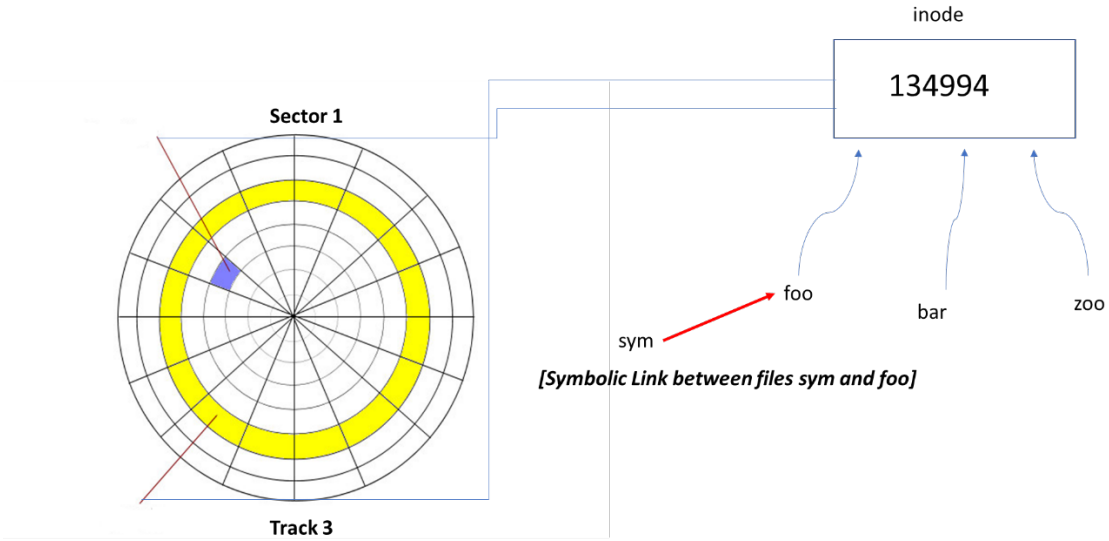


ix. All three files share the same disk storage(s) locations

x. Enter each of the following commands to verify storage content:

   a. **cat foo**
   b. **cat bar**
   c. **cat zoo**
   d. Each of these files is linked to the same storage space that contains file content "Linux is a very powerful operating system
   e. Enter the command rm foo to remove the foo file
   f. Re-issuing either the **cat bar** or **cat zoo** will still display the content stored at inode 134994
   g. Re-link foo by entering command **ln bar foo**

xi. We can also add a Symbolic Link which can serve a "short cut" to link files, for example, on different partitions. The symbolic link will point to the file, not the storage location as illustrated:



xii. Enter command **ln -s foo sym**

xiii. Enter command **ls -li**; note the difference between the link formats

xiv. Enter the following command to verity sym content

   a. **cat sym**; should be the same as foo, bar and zoo
   b. Remove the foo file and reenter the **cat sym** command
   c. Note the symbolic link has been removed and now sym is unable to reference any storage space
   d. Enter **ls -li** command
   e. Re-link foo by entering command **ln bar foo** and the sym file is reconnected

3. While still inside of the Link subdirectory, do a word count to verify.
    a. Enter command **cat foo**
    b. Enter the command
    c. **echo "Append this text to the existing text in the file foo" >> foo**
    d. Enter command **cat foo**
    e. Note the new text added to foo
    f. Enter the **wc foo** command
    g. What do these values mean? **Recall lines, words and character…**

4. Return to your Home Directory (**cd ~**); working with the grep command
    a. Enter the following command
    b. **echo "This is a grep example search file" > grepexample**
    c. **echo "grep searches content in the file" >> grepexample**
    d. **echo "add this string with the words changed and changes to the file" >> grepexample**
    e. Enter the following command; use -o switch/option to search file content and return match.
        i. **grep -o "change" grepexample;** will return the word change when found
        ii. **grep -o "change[ds]" grepexample;** will return the word changes or changed when found
        iii. Use the dmesg command to return Linux kernel memory information (akin to a kernel log file). Use with the grep search command to look for string sda. The | (pipe) symbol is located above the \ (back slash) symbol. Displays information in the buffer for device sda
            1. **dmesg | grep sda**

5. Archiving files using the TAR command
    a. Enter the following command to archive grepexample using TAR
        i. **tar -cvf oldgrepfiles.tar** grepexample
        ii. Enter **ls -l** view file listing
        iii. You should see a file with the following format **oldgrepfiles.tar**
        iv. Enter the following command to view content of archive listing for oldgrepfiles.tar
        v. **tar -tvf oldgrepfiles.tar**
        vi. **Review the tar –(switch options to become more familiar with the commands.**