CIT 371 Lab 17: Linux file types and inodes

This lab can be done with SSH/PuTTY or with the Web Console. See the Student VM Access document for information on accessing your VMs.

Log in to Coivcenter, start your VM (VM2) and log in with your own account. Open two terminal windows and in one, su to root. Note that to answer some questions, you will need to refer to the textbook. It is strongly recommended that you read chapter 10 before doing this lab.

1. inodes are data structures that store information about files, one inode per file. Including with the inode are pointers to actual disk blocks.
   1. cd to ~. Type **touch foo1 foo2 foo3**. Type **ls –i**. This option prints the inode number associated with every file. Are the inode numbers consecutive? Probably not. But they are close to each other. Delete foo1 and type **touch foo4**. Repeat your ls command. *Compare the inode number for foo1 from above to foo4 now. What might you conclude about this?*
      1. **The inode number of 10817 was first assigned to foo1, when we deleted foo1 and created foo4 the inode number 10817 is replaced to foo4. We can conclude that the inode number of foo1 was replaced with foo4 when we created and deleted the file.**
   2. As you have seen, directories are treated like files in that you can issue file commands on them. Type **ls –i /home** to view the inodes of the user directories. One user directory will have a much lower inode number than the others. *Why?*
      1. **The directory of alstottk is the one that has the lower inode value, we can expect this because of alstottk being the home directory and is one of the first directory files that inodes is assigned.**
2. Types of file entities that are not files
   1. Aside from directories, Linux treats other types of entities as files: links, devices, named pipes. Here, we explore named pipes. A named pipe is like an ordinary pipe between two instructions but it is stored in the file system (like a file). To create a named pipe, use **mkfifo**. *What does FIFO mean?* Type **mkfifo mypipe**. Type **ls –l**, *what type of file is mypipe, that is, what letter is used to describe the file type in the permissions?* To use your named pipe, you redirect the output of a program to it. Type **ls –l > mypipe**. *What happened?* Type **control+c**. *What happened?* Repeat the ls command to redirect to mypipe but add & at the end. Type **jobs** and you can see ls –l > mypipe is still running. This process is waiting for you to complete the pipe operation. Type **cat mypipe**. *What happened?* Type **jobs** again. *Is the ls command still there?* *What can you conclude about using the named pipe?* 
      1. **FIFO means first in first out. This is a special file named pipe. Mypipe is a read write file and is highlighted in black with yellow texts, the letter p is used to describe the mypipe. The p represents a named pipe. This command started to run the mypipe but continues to run since mypipe. The control c kills the command and prompts you with a new command line. Cat mypipe runs the command and does a long list. The ls command is executed, and the job is completed. We can conclude that using the named pipe is a shortcut for us to run a ls -l command.**
   2. Type **ls –l > mypipe**. Open a second terminal window, change to your home directory, and type **cat mypipe**. *What happens in your original window when you type this?* Delete mypipe using rm. *Does this operation differ from deleting a normal file using rm?* ‘’
      1. **There was a notification that popped up and said command completed and the first terminal went back to a new command line while the other terminal ran the command. This operation of deleting mypipe was normal as to deleting a file using rm.**
   3. cd to /dev. This directory stores physical and logical devices that we can interact with through file commands. Type **ls –l**. You will find links (l), directories (d) and devices that start with letters ‘b’, ‘c’, and ‘s’. *What do each of these letters stand for?* (hint: see table 10.1 on page 398). random and urandom are not physical devices but programs which act like devices in that we can redirect their output. We used urandom to generate characters for passwords back in lab 19. Type **cat urandom | head –c10**. This will display the first 100 characters returned by urandom. *What do you get (not the actual output, just describe what it looks like).* Redo this command piping cat to **tr –cd** ‘**[:alpha:]**’ **|** and then **head –c10**.
      1. **The letter b means block device, c means character device, and s means domain socket. It looks like I got a bunch of random letters and punctions marks such as ?, ; , and @ mixed in with capital letters.**
   4. The file null is a “trash can”. Repeat your cat command from the end of part c but add **> null**. Type **cat null**. *What do you get as output?* We use null to redirect output from a program that we do not actually want. For instance, many programs produce error information as a side effect. By adding > /dev/null, we force all output to the trashcan.
      1. **You get the output of nothing or deleted or moved to a trashcan.**
   5. Recall that our hard disk is called sda and that the various partitions we created in lab 18 were called sda1, sda2, sda3 and sda5. If you type ls, you will see sda1 through sda6. We can dump the contents of each of these disk partitions using cat. In order to so, we must be root. su to root and type cat sda3 | more. sda3 is the /var partition which, among other things, includes log files. You will see both ASCII text and garbage characters (which is the windows best attempt at displaying binary data). After a short while, type q to exit more. You will also notice in /dev a number of ttys. *What are these?* Notice that the sda devices are ‘b’ type and tty devices are ‘c’ type. *Why?* Exit as root.
      1. **Ttys are a command that displays terminal information such as printing the file name of the terminal and the input. The sda devices are ‘b’ for block files which is appropriate because this is our hard disk that block files will be held. Compared to ttys which are ‘c’ type meaning they are character device files which makes sense because of the display information being given about the terminal.**
3. stat provides more information about a file than ls.
   1. Return to your home directory and type **stat foo\***. This will display the stat output for the foo files you created in step 1a. *What information does stat give you? Aside from the filename itself and the inode numbers, are there any differences between the four files and if so, what?*
      1. **The information that stat gives you is such things as file name, file size, blocks, IO block, devices, inodes, links, modify, change, birth, and much more. As far as I can see there is no difference between the four files.**
   2. Type **ps aux >> foo2**. This will fill foo2 with some data. Repeat the stat command.  *Aside from the name and inode, what differences now exist between the two files’ information?* 
      1. **A lot has changed such as the size, blocks, modify, and change times.**
   3. Issue the command **stat –f /home**. This provides information on the entire directory. *How many blocks are there in this directory, how many are free and how many are available?* Notice this also gives free and total inodes. We can obtain similar information using df –k and df –i (we visit these instructions in the next lab).
      1. **There are 259584 blocks total. There are 236391 free blocks and 236391 available blocks.**
   4. The stat command can be tailored to output different types of information. Examine its man pages. The –c option is followed by one or more formatting modes. For instance, **-c “%n %A”** prints the file’s name and permissions in a human readable format. Write a stat command to output the foo\* files, displaying their name, inode number, file type, total size in bytes, and time of last change since the epoch. What command did you enter?
      1. **Stat -c “%n %i %F %s %Z” foo\***
   5. Using the tool of your choice (egrep, awk, a shell script), provide an instruction or script which will use stat to obtain all files in /etc that have at least 7 hard links. To obtain the number of hard links, use stat –c “%n %h”. The %n provides the name, the %h provides the number of hard links. *Which files were found? Include in your answers the instruction or script that you wrote, as well as a screenshot of the output.*
      1. **ls -l etc | awk '(index($1,"d") == 0) {print "etc/" $9}' | xargs stat -c "%n %h" | awk '$2 >= 2 {print $1}'**

Shut down your VM if desired, disconnect from the VPN if you are using it, and submit your lab report.