This is Lab Worksheet 8 - not an Assignment

This Lab Worksheet contains some practical examples that will prepare you to complete your Assignments. You do not have to hand in this Lab Worksheet. Make sure you complete the separate Assignments on time. Quizzes and tests may refer to work done in this Lab Worksheet; save your answers.

Before you get started - REMEMBER TO READ ALL THE WORDS

You must have an account on the Course Linux Server to do this lab. Log in to the server and use the shell. Review the Class Notes related to this worksheet as you work through it. Leave your work on the Linux server. Do not delete any work from the Linux server until the term is over and your marks are complete!

Linux File System Permissions (modes)

Commands, topics, and features covered

Use the on-line help (**man** command) for the commands listed below for more information.

- **`chmod`** – (change mode) Change the permissions (mode) on an existing inode (file, directory, etc.)
- **`ls -lid`** – (list structure, **long** version, **inode**, **directory**) See the permissions of an inode
- **`umask [`*`value`*`]`** – (user mask - shell built in) Display or change the octal **umask** value for this shell

See the course notes and man pages regarding the new commands **`umask,`** and **`chmod.`**

Log in to the Course Linux Server to do all commands in this lab. Set your bash **`PS1`** shell prompt to show your login name, computer name, and the **basename** of your current directory, just as you did in the previous Lab. **Leave your finished work on the server; do not delete it when you are finished the worksheet**.

1 Exercise: Conversion between symbolic mode and octal mode

For each **nine**-character symbolic permission, give the equivalent **three**-digit **octal** permission and the three-character **symbolic** permissions that apply to each of **User/Owner**, **Group**, and **Other**:

Row	Symbolic Mode	Octal Mode	User/Owner	Group	Other
1.	**`rwxrw-r-x`**	_	_	_	_
2.	**`r---wx-w-`**	_	_	_	_
3.	**`--x------`**	_	_	_	_

2 Exercise: Conversion between octal mode and symbolic mode

For each **three**-digit octal permission in the following table, give the equivalent **nine**-character **symbolic** permission and the three **symbolic** permissions that apply to each of **User/Owner**, **Group**, and **Other**:

Row	Octal Mode	Symbolic Mode	User/Owner	Group	Other
1.	`000`	_	_	_	_
2.	`001`	_	_	_	_
3.	`020`	_	_	_	_
4.	`300`	_	_	_	_
5.	`004`	_	_	_	_
6.	`050`	_	_	_	_
7.	`600`	_	_	_	_
8.	`007`	_	_	_	_
9.	`715`	_	_	_	_

Did you read **all the words** before completing the above exercise? Symbolic or numeric? Read all the words.

3 Exercise: Understanding directory permissions

This exercise chooses eight different permissions for a directory and then for each of the eight permissions attempts some basic commands such as **`cd`**, **`touch`**, **`mkdir`**, and **`ls`**. Different permissions will allow or deny each of these actions. You are to set the permissions, try each of the actions, and record whether or not it worked. Are the results as you would expect?

A) Execute the following three commands to create a testing directory:

```
[user@host ]$ cd ; rm -rf lab08 ; mkdir -p lab08/top ; cd lab08
```

For **each** row of the table below, repeat these next **two** steps (B and C) that will set some permissions on the **`top`** directory and then try four commands to see if those commands work with those directory permissions:

B) From in the **`lab08`** directory change the permission of the **`top`** directory using the **`chmod`** command given in the second column (**Command line**) of the table. Execute this **`chmod`** command in the **`lab08`** directory to set the permissions on the **`top`** subdirectory.

C) Next, for the row and permission value you just set in Step A, try to execute the four commands listed across the top of the table. For each of the four commands, record in the table whether or not the command line executes successfully. Enter in the table **PD** for **Permission Denied**; **OK** for **success**. The four commands to try are these (also listed across the top of the table):

1. **`cd top`** followed immediately by "**`cd ..`**" only if it **worked** (gave no error messages) You need to "**`cd ..`**" to get back up to the **`lab08`** directory only if the **`cd`** command **worked**.
2. **`touch top/file`** followed immediately by "**`rm top/file`**" if it **worked**
3. **`mkdir top/dir`** followed immediately by "**`rmdir top/dir`**" if it **worked**
4. **`ls -l top`**

- You will carry out Steps B and C above (one **`chmod`** and four other commands) for **each** of the rows of the table below. Do the **`chmod`**, then try each of the four commands and record **PD** or **OK**.
- Before you run each **`chmod`** command in the table below, ensure your current directory is **`lab08`**.
- If the **`cd`** into **`top`** **works** (no error), follow it immediately with "**`cd ..`**" to return up to the **`lab08`** directory again, otherwise you will be in the wrong directory for the next command in the table.
- If the **`touch`** **works** (no error), immediately remove the file you just created.
- If the **`mkdir`** **works** (no error), immediately remove the directory you just created.
- If you get any error other than **Permission Denied**, you either made a typing error or your current directory is not **`lab08`** – you probably forgot to do "**`cd ..`**" to return up to the **`lab08`** directory again after a successful **`cd top`** command. You must always work in the **`lab08`** directory.

Table of OK and PD for different values of chmod permissions					
Row	**Command line**	`cd top`	`touch top/file`	`mkdir top/dir`	`ls -l top`
1.	`chmod 000 top`	_	_	_	_
2.	`chmod 100 top`	_	_	_	_
3.	`chmod 200 top`	_	_	_	_
4.	`chmod 300 top`	_	_	_	_
5.	`chmod 400 top`	_	_	_	_
6.	`chmod 500 top`	_	_	_	_
7.	`chmod 600 top`	_	_	_	_
8.	`chmod 700 top`	_	_	_	_

Did you read **all the words** before completing the above exercise? Your table must contain only **OK** or **PD**. If you see any other error such as "**no such file or directory**", you are either typing incorrectly or you are not in the correct **`lab08`** directory when you are running that command.

4 Exercise: Minimum Permissions for common file operations

In the table below, give in **symbolic `rwx`** form the **minimum** permissions an ordinary (non-root) user requires to successfully complete the commands listed in the left column below. "**Minimum**" means the *least* amount of **`rwx`** permissions needed. How many of **`rwx`** can you *take away* and still perform the given command *successfully*? What are the **minimum** permissions needed? Test your guess to make sure you are correct.

Recall that directories store only names and inode numbers, not data. Recall that some commands require permissions **only** on the directory inode; some require permissions on the directory **and** the data inodes.

```
1. [user@host]$ cp       srcdir/srcfile  targetdir/   (new file name and content)
2. [user@host]$ mv       srcdir/srcfile  targetdir/     (rename to new file name)
3. [user@host]$ ln       srcdir/srcfile  targetdir/           (add new file name)
4. [user@host]$ rm       srcdir/srcfile            (remove an existing file name)
5. [user@host]$ cat      srcdir/srcfile                 (display an existing file)
6. [user@host]$ date  >> dir/oldfile                   (append to an existing file)
7. [user@host]$ date   > dir/newfile        (create a new file in an existing directory)
```

For commands 6 and 7 above, the directory is not a "*source*" directory since nothing is being read from it, and the file is not a "*source*" file since it is being written to (it is an output **target**). Use the first two columns in the table below to record permissions for the directory and the target file for 6 and 7 above.

Table of MINIMUM `rwx` symbolic permissions needed to perform each of the above commands			
Command used	**on the (source) directory**	**on the (source) file**	**on the target directory**
1. copy a file	_	_	_
2. move a file	_	_	_
3. link to a file	_	_	_
4. delete a file	_	_	**N/A**
5. read a file	_	_	**N/A**
6. append to an existing file	_	_	**N/A**
7. create a new file	_	**N/A**	**N/A**

Now go back and re-read all the words of this question, including the word "**symbolic**" in the first line. Make sure all the answers are the **minimum** permissions needed to do the given operation. (You could not remove any more of the given permissions and still have the operation work.)

5 Shell umask and permissions for new files and directories

The shell **umask** value restricts the permissions given to **new** files and directories when they are **first created**. Without the **umask** value, or with a zero value, a new file would be created with full permissions **666** and a new directory with full permissions **777**. The **umask** *masks* out permissions - each bit of the **umask** turns **off** the corresponding permission in the newly created file or directory. A **umask** value of of **777** turns off **all** permissions on new files and directories; a **umask** of **000** doesn't turn off any permissions (not recommended!).

Note that **masking** is *not* the same as **subtracting**, e.g. `666` masked with `001` is still `666` and `666` masked with `003` is `664`. The mask turns **off** permission bits. If they are already off, the **umask** makes no change:

- **`rw-`** (**6**) ***masked*** with **`--x`** (**1**) is **`rw-`** (**6**) because the **`x`** bit in the mask does not change any permissions
- **`rw-`** (**6**) ***masked*** with **`-wx`** (**3**) is **`r--`** (**4**) because only the **`w`** bit is changed (turned off) by the mask

Each process, and thus each shell, has its own **`umask`** value. Different Linux distributions set different default (at login) **`umask`** values. The values in your particular distribution of Linux may not be the same as other distributions. The values set by the system administrator may differ from the distribution defaults. Do not rely on the **`umask`** having any standard value.

a) What built-in shell command displays or sets the **`umask`** value: __________________
b) Give the default (at full login) **`umask`** for your account: _______________

For each row of the following table **set** the given **umask** and then fill in the four new permission columns:

c) **Set** and use the **given umask** in column two before filling in the permissions in the rest of the row.
d) Based on the **umask** in column two, write down the **permissions** that would be **given** to a new **file** and a new **directory**. Give both the **symbolic** and **numeric** forms. You can create a new file and a new directory to verify your answer, but you should calculate and know the answer without needing to create anything. Using the given **umask**, what would be the permissions set on a **new** file and a **new** directory?

Row	**Set your umask to this value and then fill in the fields on the right:**	**create *new* file permissions**		**create *new* directory permissions**	
		symbolic	**numeric**	**symbolic**	**numeric**
1.	**`umask 0001`**	_	_	_	_
2.	**`umask 0002`**	_	_	_	_
3.	**`umask 0003`**	_	_	_	_
4.	**`umask 0022`**	_	_	_	_
5.	**`umask 0077`**	_	_	_	_
6.	**`umask 0777`**	_	_	_	_

After you are finished the above exercise, exit your shell or reset your **umask** back to the normal **umask**. Do **not** continue to use a shell that has a **umask** of **0777**.

1. **True or False** - the **`umask`** can change the permissions of **existing** files and directories? ___________
2. What command name changes the permissions of **existing** files and directories: _____________
3. What value **`umask`** gives the **owner** and **group** full permissions on new files and directories: _______
4. What value **`umask`** gives only the **owner** full permissions on new files and directories: _________