Student Name: _____ **Lab Section:** _____ **/3** marks

Due Date - Upload to Blackboard by 8:30am Monday February 6, 2012

Submit the completed lab to Blackboard following the **Rules for submitting Online Labs and Assignments**.

Before you get started...

Refer back to Lab Two, *Using Standard Linux Commands I,* and ensure you understand everything written in the two sections near the beginning named **Linux Absolute and Relative Pathnames** and **Linux shell command syntax**. Log in as the **non-root** user and open a **non-root** shell terminal window to do this lab.

To explore the directory structure, the following commands will be used:

- **`man`** - to get help for commands or system files or topics
- **`cp`** - copy one file to another, or copy one or more files into a directory
- **`mv`** - move/rename pathnames, or move multiple pathnames into a directory
- **`rm`** - delete files (and entire directories of files recursively, with the **`-r`** option)
- **`cat`** - display the contents of files without pagination (usually onto your screen)
- **`less`** (also **`more`**) – to page through a text file one screenful at a time (better than **`cat`**)
- **`passwd`** - to change a password (usually your **own**; only **root** can change others)
- **`find`** - to find pathnames (e.g. files or directories) by name, or by userid, or other criteria
- **`touch`** - to create an empty file and/or to update the file's date/time modified stamp
- **`clear`** - to clear the screen of a terminal and put the cursor back at the top of the screen

Log in as the **non-root** user and open a shell **terminal** window to do **all** commands in this lab. Do not use **root**!

Online help: Man pages revisited

The **`man`** command, short for Manual Pages, displays the manual page for the specified command. Man pages, as they are commonly referred to, contain all of the pertinent information on the basic command concepts, how to use the command, the command structure, basic options available for the command and how to use them, advanced options (if any), and related topics, in that order. The **`man`** command syntax is:

- **`man`** ***`command`*** - where ***`command`*** is the name of the command or thing you wish to learn about.

Examples: **`man ls ; man man ; man passwd ; man group ; man hosts`**

A text screen will show up with the information you requested - if it exists. You can then scroll up and down the man page using the **`up`** (↑) and **`down`** (↓) arrow keys and/or the **`[PgUp]`** and **`[PgDn]`** keys on your keyboard. You can also use the **spacebar** to scroll down one **screen**. Once you are done with the man page, simply type **`q`** for quit and you will exit the man page display. You can type **`q`** any time you want to exit the manual pages and you can type **`h`** or **`?`** for a **help** screen listing all the **other neat things** you can do while looking at this manual page. The most common thing to type is a **blank** (space), which displays the **next page** of the help file.

When you don't know the specific command for an operation, you can search the man page **titles** based on a keyword. (You can only search the title lines.) For this you need to specify the **`-k`** (keyword) option:

- **`man -k`** ***`keyword`*** (*multiple keywords will find more title lines for each word)*

Example: **`man -k games`** **(***lists all man page title lines that contain the word* ***games)***

1 Command: touch

The **touch** command updates the "last modified" time/date stamps on one or more existing files. It can also be used to create one or more **new**, **empty files**. See the manual page for more features.

Creating empty files and updating the modification time

Perform the following commands shown in **bold** type. Most commands will produce no output if they succeed.

```
[user@host  ] $ cd
[user@host ~] $ rm -rf lab3.1                    (remove this directory and everything inside it)
          (The above command will make a "clean slate" if you choose to restart this section from the start.)
[user@host ~] $ mkdir lab3.1                     (create a new, empty sub-directory)
[user@host ~] $ cd lab3.1                        (make this the new current working directory)
[user@host lab3.1] $ touch clock                 (create a new, empty file)
[user@host lab3.1] $ ls -li clock        (Note that -l is lower-case letter L, not the digit 1)
```

a) Record **only** the **index** number and **time/date** stamp: _____

```
[user@host lab3.1] $ sleep 60          (Waits for one minute – 60 seconds.  Read a book.)
[user@host lab3.1] $ touch clock       (update the time stamp on the existing file)
[user@host lab3.1] $ ls -li clock      (that -l option is a letter, not a digit)
```

b) Record **only** the **index** number and **time/date** stamp: _____

2 Command: cp (copy)

The **cp** (Copy) command makes a copy of files or directories. The syntax for the **cp** command is:

- **cp [*options*] *sources...* *destination***

where ***sources...*** is one or more files or directories and ***destination*** is either a file or a directory. If the destination is a directory, the file(s) will be copied into that directory using their **same names**. If you want to copy **directories**, you **must** use options such as **-r** or **-a**; otherwise, **cp** copies only source **files**.

```
[user@host  ]$ cd
[user@host ~]$ rm -rf lab3.2             (remove this directory and everything inside it)
[user@host ~]$ mkdir lab3.2
[user@host ~]$ cd lab3.2
[user@host lab3.2]$ touch a b c
[user@host lab3.2]$ ls -i
```

a) Give the output of the last command, above: _____

```
[user@host lab3.2]$ mkdir mydir
[user@host lab3.2]$ ls -F                (that is an UPPER CASE option letter)
```

b) Give the output of the last command, above: _____

```
[user@host lab3.2]$ cp a b c mydir
[user@host lab3.2]$ ls -i mydir
```

c) Give the output of the last command, above: _____

```
[user@host lab3.2]$ mkdir snack
[user@host lab3.2]$ touch snack/pie
[user@host lab3.2]$ cd snack
[user@host snack]$ touch apple
[user@host snack]$ cd ..
[user@host lab3.2]$ cp snack/pie snack/apple mydir        (mydir was created earlier)
[user@host lab3.2]$ ls mydir
```

d) Give the output of the last command, above: _____

```
[user@host lab3.2]$ mkdir A B C          (note that these are UPPER CASE directory names)
```

e) What **command line** could you use to verify that the **three** directories have been **created?**

_____

```
[user@host lab3.2]$ touch  A/foo  B/bar  C/empty    (create three empty files)
[user@host lab3.2]$ cp A B C mydir          (try to copy A and B and C into mydir)
```

f) Record **one** of the messages displayed on the screen: _____

```
[user@host lab3.2]$ ls mydir                    (confirm that no directories were copied)
```

g) Why were the three source **A,B,C** directories **not copied** into destination directory **mydir**?

_____

```
[user@host lab3.2]$ cp -r A B C mydir          (the copy now succeeds using this option!)
[user@host lab3.2]$ ls -R mydir                  (that is an UPPER CASE option letter)
```

h) What **command line** could you use to see the **index number** and **date** of the **new copy** of file **empty**?

_____

```
[user@host lab3.2]$ mkdir -p parent/child                  (remember what -p does?)
[user@host lab3.2]$ cp -r --parents parent/child mydir          (read the man page)
```

i) Give the **absolute path** of the **new copy** of directory **child** after the above copy command creates it:

_____

3 Command: mv (move or rename)

The **mv** (Move or Rename) command moves (renames) files or directories. The syntax for the **mv** command is:

- **mv [*options*] *sources...* *destination***

where ***sources...*** is one or more files or directories and ***destination*** is either a file or a directory. If the destination is a **directory**, the source files or directories will be moved (renamed) into that directory using their **same names**. If the destination is a **file**, only **one** source file is allowed to be moved (renamed). Examples:

- **mv file1 newfilename1**
- **mv directory1 newdirectoryname1**
- **mv file1 directory/newfilename1**
- **mv file1 file2 file3 directory**

```
[user@host  ]$ cd
[user@host ~]$ rm -rf lab3.3              (remove this directory and everything inside it)
[user@host ~]$ mkdir lab3.3
[user@host ~]$ cd lab3.3
[user@host lab3.3]$ touch A                          (create a single new, empty file)
[user@host lab3.3]$ ls -i                              (note the index number)
[user@host lab3.3]$ cp A foo                             (copy the file)
[user@host lab3.3]$ mv A bar                             (move the file)
[user@host lab3.3]$ ls -i                             (note the index numbers)
```

a) Explain why **foo** and **bar** have different index numbers: _____

_____

```
[user@host lab3.3]$ touch green blue orange          (create three new empty files)
[user@host lab3.3]$ mv green blue orange     (try to move two files into a third file)
```

b) Record the error message: _____

```
[user@host lab3.3]$ mkdir colours
[user@host lab3.3]$ mv green blue orange colours
```

c) Give the **absolute** path of the file **blue**: _____

```
[user@host lab3.3]$ mkdir fans players arena      (create three new, empty directories)
[user@host lab3.3]$ touch  fans/me  players/you          (create two new, empty files)
[user@host lab3.3]$ mv fans players arena    (move two directories into a third directory)
```

d) Give the **absolute path** of the file **you after** the above **mv** command has moved it:

_____

4 Command: rm (remove or delete)

The **rm** (Remove or Delete) command removes (deletes) **files**. If the **-r** option is specified, it recursively deletes **directories** and **all** their contents. Unlike DOS, Windows, or OSX, a file or directory that is deleted with the **rm** command is **gone** (is **not** saved in a Recycle Bin) and not easily recovered. The syntax for the **rm** command is:

- **rm [*options*] *pathnames...***

Another useful option to **rm** is **-f** (force) that turns off any interactive prompts and most error messages. (We have been using "**rm -rf**" to completely recursively remove lab directories at the start of each section.)

Note: Most Unix/Linux shells let you type multiple commands on one line by separating them using the semi-colon character ';'. Type the **four** commands below, separated by **three** semi-colon characters:

```
[user@host  ]$ cd ; rm -rf lab3.4 ; mkdir lab3.4 ; cd lab3.4
[user@host lab3.4]$ mkdir sandbox sandbox/toddlers          (create two directories)
[user@host lab3.4]$ touch child1 child2 child3              (create three empty files)
[user@host lab3.4]$ mv child1 child2 child3 sandbox/toddlers  (move all 3 files)
[user@host lab3.4]$ ls sandbox/toddlers                  (you should see three child files)
[user@host lab3.4]$ rmdir sandbox                 (try to remove the non-empty directory)
```

a) Record the error message: _____

```
[user@host lab3.4]$ rmdir -p sandbox          (try again to remove the non-empty directory)
```

b) Record the error message: _____

```
[user@host lab3.4]$ cp -a sandbox savebox    (save a full copy of sandbox in savebox)
[user@host lab3.4]$ ls                       (confirm that is sandbox has been copied to savebox)
[user@host lab3.4]$ rm -r sandbox            (recursively delete sandbox and everything in it)
[user@host lab3.4]$ ls                       (confirm that sandbox is gone)
[user@host lab3.4]$ mv savebox sandbox       (what does this do?)
```

c) Give the **absolute path** of the file **child2 after** the above commands are finished:

_____

```
[user@host lab3.4]$ cp -a sandbox/toddlers sandbox    (recursive copy FAILS – why?)
```

d) Explain why the above copy fails: _____

_____

The **-i** option to **rm** will turn on "Interactive" mode, where you are prompted about every file being deleted:

```
[user@host lab3.4]$ cp -a sandbox/toddlers .    (note the DOT ending this command line)
[user@host lab3.4]$ ls toddlers                 (you should see three child files in here)
[user@host lab3.4]$ rm -ri toddlers             (answer yes to all the interactive questions)
```

5 Command: cat (catenate or show contents)

cat (Catenate, or Show) opens one or more files and catenates (shows) their contents. You can use it on any size file, but files are not be paginated and large files will scroll off your screen. Unlike **less**, **cat** won't warn you if you're about to **mess up** your terminal by displaying a **binary** format file. The syntax for **cat** is:

- **cat [options]** ***file_list***

Try these examples using **cat** and **less**:

- `[user@host ~] $ cat /etc/issue`
- `[user@host ~] $ cat /etc/fstab`
- `[user@host ~] $ cat /etc/issue /etc/fstab`
- `[user@host ~] $ cat /etc/services` (over 10,000 lines!)
- `[user@host ~] $ cat /etc/services | less` (a long way of doing the next command)
- `[user@host ~] $ less /etc/services` (the right way to use **less**)

a) What option to **cat** shows non-printing characters? _____

b) What option to **cat** will number the output lines? _____

c) What option to **cat** will suppress repeated empty output lines? _____

6 Command: clear

```
[user@host ] $ clear
```

a) What is the purpose of the **clear** command? _____

7 Command: passwd (change your password)

The **passwd** (Password) command changes user account passwords. The **root** *super-user* can change any user account password; ordinary users can only change their own passwords. The command may verify that any password you choose is a secure password - i.e. that it is not a simple known dictionary word and that it is long enough to be secure. A good, secure password should be no less than 6 alpha-numeric characters in length, and contain at least one special/numeric character within it. Note: **None of the characters you type for your password will echo on your screen, for security. You will be typing blind.** To change your password follow the steps outlined below (the password will not echo on your screen as you type it):

`[user@host ]$ passwd` (only **root** can supply a user name argument)

1. The default is to change the **current user** password; **root** can supply one user name as an argument..
2. The command asks you for your current password, to confirm you really ARE you.
3. It will then ask you for a new password. Type the new password. (**Your typing will not echo**.)
4. If the new password is acceptable, it will then ask you to retype it to confirm; otherwise, you'll need to pick a better password..
5. If the operation was successful the **passwd** utility displays a message indicating that it was.

8 Command: find (find pathnames)

The **find** command recursively walks the directory tree structure, starting at a pathname given by the user, and finds other pathnames based on many optional criteria. See the man page for the many options and features. The most common use is to find pathnames containing a particular **basename** pattern:

- **find *starting_directories...* -name '*basename*' -print**

This will print any pathname ending in the ***basename*** pattern, found in any directory, starting from each of the the ***starting_dirctories.*** Note the unusual use of **full-words** as **options** in this command. The ***basename*** patterns can include shell-GLOB-style path metacharacters such as "*****" and "**?**". Examples:

- **find /etc -name 'passwd' -print** (no GLOB metacharacter [wildcard] characters)
- **find /etc -name '*.conf' -print** (all paths ending in **.conf**)
- **find /bin -name '?ash' -print** (paths ending in four-character names ending in '**ash**')
- **find /lib /usr/lib -name 'lib*.a' -print** (multiple starting directories)
- **find . -print** (prints **all** the pathnames under the current directory)
- **find . ! -user root -size +10 -mtime -3 -type f -print** (see the man page!)

a) What does **-user** mean, above? _____

9 Review exercise

Enter exactly the commands that are shown in **bold** below and note which commands produce **errors**. (There will be **three** errors; this is intentional.) Answer the questions following based **only** on these **review** commands. The **tilde** characters below have the same meaning as in the previous lab. (Go look!) Be precise in your typing!

1. **cd ; rm -rf ~/3rev**
2. **mkdir ~/3rev**
3. **cd ~/3rev**
4. **mkdir ./orchard**
5. **touch apple orange**
6. **mv orange orchard/lemon**
7. **rm orange**
8. **touch lettuce tomato cucumber**
9. **cp tomato lettuce garden**
10. **mkdir jardin forest**
11. **mv lettuce cucumber jardin**
12. **rmdir garden**
13. **touch lab3**
14. **cd orchard**
15. **cd ../../3rev/forest**
16. **mv ../lab3 ../tomato**

a) Give the command **number** that generated the error followed by the **full** and **exact** error message:

b) What is the **absolute** path of the shell's current working directory after the **last** command, above?

c) Give the **absolute pathname** of the one regular file **`lemon`** that is now in the directory named **`orchard`**:

d) Give the relative path to the same **`lemon`** file from the **`forest`** directory:

e) Give the relative path to the same **`lemon`** file from your own **HOME** directory:

f) Give the relative path to the same **`lemon`** file from the directory called **`/home`**:

g) Give the relative path to the same **`lemon`** file from the Linux ROOT directory:

h) Give the relative path to the same **`lemon`** file from the directory called **`/root`**:

i) List the **basenames** of **directories** that were successfully created (at any time) during the **review** exercise:

j) List the **absolute pathnames** of all **directories** that were **successfully** deleted during the **review** exercise:

k) List the **absolute pathnames** of the five **regular files** still remaining anywhere under the directory **`3rev.`** Do **not** include the names of any **directories** or sub-directories – list only the five absolute **regular** file names located **anywhere** under the review directory **`3rev`**:

l) What command line recursively **finds** and displays **all** pathnames under **`~/3rev`** ?

m) What command line recursively **finds** and displays **all** pathnames owned by **your userid** under the system directory **`/var/spool`** (you will see many **Permission denied** messages in the output)?

n) Did you **READ *ALL* THE WORDS** in this lab? (Yes/No) _____