This is Lab Worksheet 2 - 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!**

Commands introduced and used in this Lab

- **`PS1='[\u@\h \W]$ '`** set shell prompt to include *user*, *hostname*, and **basename** of **`pwd`**
- **`cd`** – change the **current working directory** of the shell
- **`find`** **-** recursively find pathnames (e.g. files or directories) by name, userid, date, or other criteria
- **`less`** (also **`more`**) – to **paginate** output on your screen, one screenfull at a time (used by **`man`**)
- **`ls`** – "List Structure" – list directory content (what pathnames are in a directory)
- **`man`** – read the manual page (help file) for a command using the built-in pagination program "**`less`**"
- **`mkdir`** – create one or more new empty directories
- **`passwd`** **-** to change a password (usually your own; only **root** can change others)
- **`pwd`** – print the absolute pathname of the **current working directory** of the shell
- **`rmdir`** – remove one or more **empty** directories (use a variation of **`rm`** to remove **non-empty** ones)

1 *Command:* `man`

The **`man`** (Manual) command takes the **name** of a command as a parameter, e.g. "**`man pwd`**" or "**`man ls`**". It displays the first page of a help file and **pauses**, waiting for you to type "**`q`**" to quit reading or "**`h`**" for more options. The most common thing to type is a **blank** (space), which displays the **next** page of the help file.

a) Read the man page for the **`pwd`** command and give its full **NAME** (one-line description) here:

____

Use the **`man`** command to read up on each of the commands you use in this course, including the **`man`** command itself ("**`man man`**"). The **`cd`** command is **built-in** to the shell and does not have its own man page - see the man page for the **`bash`** shell for details on all built-in shell commands.

The answers to the following questions are under **DESCRIPTION** inside "**`man man`**":

b) What does **bold text** mean in the **SYNOPSIS** section of a man page?

____

c) What does ***italic text*** mean in the **SYNOPSIS** section of a man page?

____

d) What do square brackets **`[]`** mean in the **SYNOPSIS** section of a man page?

____

e) What does the pipe symbol **`|`** (**SHIFT-**\) mean in the **SYNOPSIS** section of a man page?

____

f) What do three dots (ellipsis) **`...`** mean in the **SYNOPSIS** section of a man page?

____

2 *Commands:* `cd` *and* `pwd`

Set your shell prompt: Before doing this lab, set your bash shell prompt to show your login name, the computer name, and the **basename** of your current working directory using this command that sets the **`PS1`** variable that contains the prompt (type this **exactly** and use **single** quotes and two blanks, one near the end):

- `bash-4.2$` **`PS1='[\u@\h \W]$ '`** *(two spaces; one just before the closing single quote)*
- `[`*`user@host`*` ~]$` **`echo It Worked!`** *(the **`user`** and **`host`** will be your own)*

The ***`user`*** string in the shell prompt will be your *own* **userid**, which is why it is shown in the ***italic*** font in this Lab. Text in ***italic*** font is **not** to be typed literally. The ***`host`*** string will be the hostname of the computer; it is also shown in ***italic*** font in this Lab. The shell will replace the characters **`\W`** (upper-case **`W`**!) by the **basename** of your current directory.

The **`cd`** (Change Directory) command allows you to navigate through the Linux directory hierarchy structure by changing your shell's **current working directory**. The syntax for **`cd`** is:

> ➢ **`cd [`*`directoryname`*`]`**

Typing **`cd`** with **no *`directoryname`*** argument will take you to your personal **HOME** directory (which is **not** the same thing as the directory called **`/home`** - be careful!). Providing a single ***`directoryname`*** parameter will change your shell's **current working directory** to the given directory. While you are working with the **`cd`** command, watch the shell prompt; it will change to display the **basename** of the current working directory after each **`cd`** command. Your **HOME** directory is indicated in the shell prompt by a tilde character: **`~`** This tilde character indicates you are in your own personal **HOME** directory (**not** the system directory called **`/home`** - be careful to distinguish between your **HOME** and the system directory).

a) At the command prompt type **`cd`** without any parameters. Record here the directory **basename** shown at the **right end** of the bash shell prompt: **`[`*`user@host`*** ________________ **`]$`**

b) Type **`pwd`** at the prompt and record the output here: ________________________________

c) **`cd /`** This will change the current directory to the top-level "**ROOT**" directory. What directory **basename** is shown in the bash prompt after this command?
`[`*`user@host`* ________________ **`]$`**

d) Give the output of the **`pwd`** command now: ________________________________

e) **`cd /etc`** What directory **basename** is shown in the bash prompt after this command?
`[`*`user@host`* ________________ **`]$`**

f) Give the output of the **`pwd`** command now: ________________________________

g) **`cd ..`** (Two periods.) This command will "go up" one directory level (to the **ROOT**). What directory **basename** is shown in the bash prompt after this command?
`[`*`user@host`* ________________ **`]$`**

h) Give the output of the **`pwd`** command now: ________________________________

i) **`cd home/`*`user`*** Replace ***`user`*** with the **userid** that you are logged in with now. What directory **basename** is shown in the bash prompt after this command?
`[`*`user@host`* ________________ **`]$`**

j) What is full **absolute** path of the relative path directory argument of the command from (i) above? Answer: ________________________________

k) Give the output of the **`pwd`** command now: ________________________________

l) **`cd /usr/local/bin`** What is the **basename** in the bash prompt after this command?
`[user@host` __________________ **`]$`**

m) Give the output of the **`pwd`** command now: ____

n) **`cd ../../sbin`** What is the **basename** in the bash prompt after this command?
`[user@host` __________________ **`]$`**

o) Give the output of the **`pwd`** command now: ____

p) **`cd ../local/bin`** What is the **basename** in the bash prompt after this command?
`[user@host` __________________ **`]$`**

q) Give the output of the **`pwd`** command now: ____

r) **`cd ../../bin`** What is the **basename** in the bash prompt after this command?
`[user@host` __________________ **`]$`**

s) What is the full **absolute** path of the relative path directory argument of the command from (r) above?
Answer: ____

t) What is the output of the **`pwd`** command now: ____

u) **Describe** the **effect** of executing a **`cd`** command without **any** arguments; **explain** what happens:
Answer: ____

3 *Command:* `ls`

The **`ls`**, or List Structure (list directory contents) command lists the names and/or properties of pathnames. Use it to see the names and attributes of directories and files and directories inside directories. The syntax is:

- **`ls [-options...] [pathnames...]`**

Read the man page for **`ls`** to discover many useful options that allow you to display the contents of a directory in many formats. Two common options are **`-a`** to show **all** files (including **hidden** files that start with a **leading period**) and **`-l`** (lower-case letter **L**) to get a **long** listing including most file **attributes**, such as file **owner**, file **modify** date, and file **permissions**. **Single** option letters can be typed separately or **bundled** together after a **single** dash in most Linux commands, as follows:

- **`ls -a -l [pathnames...]`** *(The option `-l` is lower-case letter `L`, not the digit `1`)*
- **`ls -la [pathnames...]`** *(The option `-l` is lower-case letter `L`, not the digit `1`)*

Perform the following commands and observe how the output of `ls` changes:

a) **`ls /bin/ls`**
b) **`ls -l /bin/ls`** *(The option `-l` is lower-case letter `L`, not the digit `1`)*
c) **`ls -lis /bin/ls`**
d) **`ls /home/user`** *(Replace **`user`** with your current login userid)*
e) **`ls -a /home/user`** *(Replace **`user`** with your current login userid)*
f) **`ls -al /home/user`** *(Replace **`user`** with your current login userid)*
g) **`ls -la /home`**
h) **`ls -ld /home/user`** *(Replace **`user`** with your current login userid)*
i) Look up the meaning of the **`-d`** option to **`ls`** in the manual page for **`ls`**. Explain what it does:
Answer: ____
j) Look up the meaning of the **`-i`** option to **`ls`** in the manual page for **`ls`**. Explain what it does:
Answer: ____

Shell file name completion: Without using the **`[Enter]`** key, type just the six characters "**`ls /ho`**" and then press the **`[Tab]`** key. The shell will fill in the rest of the "**`/home`**" name for you. Let the shell do your typing for you! Also try this pathname: **`ls -ld /lo[Tab]`**

After typing all the above commands, press the '*up arrow*' and then '*down arrow*' keys to scroll up and down in the list of commands you have typed. Note how you can re-execute any command by scrolling to it with the arrow keys and pushing the [**Enter**] key anywhere in the command line to execute it again.

Sending long output into the pagination commands *less* or *more*

Often, a directory listing might be longer than a single screen and may scroll off the top of the window you are using. You can view any long output one screen at a time using a **pagination** command such as "**less**" or "**more**". To send the output of **ls** into the input of "**less**" or "**more**", separate the commands using the "**pipe**" symbol "**|**" (found above the backslash on most keyboards). Try these three command lines:

a) **ls -al /usr/bin** *(This will produce thousands of lines of output on your screen!)*
b) **ls -al /usr/bin | less** *(This paginates the huge output one screen at a time.)*
c) **ls -al /usr/bin | more** *(This paginates the huge output one screen at a time.)*

Use the **[spacebar]** to jump to the next screen of information and the letter **b** to go backward one screen, just as you did using the **man** command. You can use **q** to **quit** the command and the letter **h** to bring up a screen of other useful commands. The **man** command uses **less** to paginate manual pages. The command "**more**" is an older version of "**less**" with fewer features - type **h** to get help as well.

4 Command: mkdir

The **mkdir** (Make Directory) command allows you to create one or more new, empty directories (folders), provided the names aren't already being used. The syntax for the **mkdir** command is:

➢ **mkdir** ***directory...***

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

```
[user@host ~]$ cd
[user@host ~]$ rm -rf lab2.4                (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 lab2.4                          (create a new, empty sub-directory)
[user@host ~]$ cd lab2.4                        (make lab2.4 the current directory)
[user@host lab2.4]$ mkdir dir1 dir2          (create two new, empty sub-directories)
[user@host lab2.4]$ ls -i
```

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

```
[user@host lab2.4]$ cd dir1              (make dir1 the current working directory)
[user@host dir1]$ ls -ia
```

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

```
[user@host dir1]$ mkdir subdir                      (create a new, empty sub-directory)
[user@host dir1]$ ls -ia
```

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

```
[user@host dir1]$ cd ..                   (two periods: go up one directory level)
[user@host lab2.4]$ mkdir parent/child              (fails to create a new directory)
```

d) Record the error message: ____
e) **Explain why** the above command **failed** and did not execute as expected:

____

```
[user@host lab2.4]$ mkdir –p parent/child             (see the man page for mkdir)
```

f) The above command **succeeds** with no errors. What does the **`-p`** option to the **`mkdir`** command do?

____

5 Command: `rmdir`

The **`rmdir`** (Remove Directory) command allows you to remove one or more directories, but only if each directory is empty (contains no files or other sub-directories). The syntax for the **`rmdir`** command is:

➢ **`rmdir`** ***`directory...`***

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

```
[user@host ]$ cd
[user@host ~]$ rm -rf lab2.5              (remove this directory and everything under it)
[user@host ~]$ mkdir lab2.5                        (create a new, empty sub-directory)
[user@host ~]$ cd lab2.5                          (make lab2.5 the current directory)
[user@host lab2.5]$ mkdir dir1 dir2 test         (create three new, empty directories)
[user@host lab2.5]$ ls -il              (option -l is lower-case letter L, not the digit 1)
```

a) Give the 4-line output of the last command, above: ______________________________

____

____

____

```
[user@host lab2.5]$ rmdir test
[user@host lab2.5]$ ls
```

b) Give the two-word output of the last command, above: ______________________________

```
[user@host lab2.5]$ mkdir  -p  dir1/subdir  parent/child
[user@host lab2.5]$ cd dir1
[user@host dir1]$ rmdir dir2                         (this fails with an error message)
```

c) Record the error message: ____

d) **Why** did the command generate this error message? Explain **why** the command failed:

____

```
[user@host dir1]$ rmdir ../dir2
[user@host dir1]$ cd ../dir2                         (this fails with an error message)
```

e) Record the error message: ____

```
[user@host dir1]$ cd ..                   (two dots means go up one directory level)
[user@host lab2.5]$ rmdir dir1/subdir
[user@host lab2.5]$ rmdir dir1
[user@host lab2.5]$ ls -il
```

f) Give the 2-line output of the last command, above: ______________________________

____

```
[user@host lab2.5]$ rmdir  parent/child  parent
```

g) **Why** doesn't the above command produce an **error message** about the non-empty directory **`parent`**?

____

6 *Review exercise:* `cd, mkdir, rmdir`

Enter the 13 commands that are shown in **bold** below and note which commands produce **errors**. (There will be some errors, this is intentional.) In the following questions, **record** the errors along with the **number** of the command line that generated each. Note the use of leading **tilde** characters below, indicating to the shell that this pathname starts in your **HOME** directory (not the directory called **`/home`**). In this case, the leading **tilde** on the pathname is shell short-hand for **`/home/`*`user`***, where ***`user`*** is your login userid.

0. **`rm -rf ~/lab2.6`** *(Note the use of the tilde character!)*
1. **`cd`**
2. **`mkdir ~/lab2.6`** *(Note the use of the tilde character!)*
3. **`cd lab2.6`**
4. **`mkdir ./hockey`**
5. **`mkdir soccer football`**
6. **`rmdir ~/lab2.6`** *(Note the use of the tilde character!)*
7. **`rmdir hockey`**
8. **`mkdir ~/lab2.6/course`** *(Note the use of the tilde character!)*
9. **`cd ..`**
10. **`cd hockey`**
11. **`cd lab2.6/football`**
12. **`rmdir ~/lab2.6/course`** *(Note the use of the tilde character!)*

Answer these questions below based **only** on commands **1** to **12**, above (ignore the first **`rm`** command):

a) Record **exactly** each error message along with the command **number** that generated the message:
____
____
____

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

c) What **command** could you use to **verify** your previous answer ? ____________________________

d) List by **basename** only all the directories that you **successfully** deleted: ______________________
____
____

e) List by **absolute pathname** every directory you **successfully** created (including ones you removed):
____
____
____
____
____

f) List every directory and sub-directory remaining under and including **`lab2.6`** using a **relative** path relative to your **HOME** directory (the **relative** pathnames must each start in your **HOME** directory):
____
____
____
____

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.

➢ **`passwd`** ***`[userid]`*** *(only root can supply a user name argument)*

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.**

- The default is to change the **current user** password; **root** can supply one user name as an argument..
- The command asks you for your current password, to confirm you really ARE you.
- It will then ask you for a new password. Type the new password. (**Your typing will not echo**.)
- 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..
- 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 (and usually prints) pathnames, based on *many* optional criteria. See the man page for the *many* options and features. The most four most common uses are given below: (a) to find *all* pathnames under a starting directory, (b) find pathnames containing a particular **basename** pattern under a starting directory, (c) find files *owned* by a particular userid, or (d) find files *modified* within some number of days:

```
(a) find  [starting_directories...]  -print
(b) find  [starting_directories...]  -name 'basename'  -print
(c) find  [starting_directories...]  -user 'userid'    -print
(d) find  [starting_directories...]  -mtime -days      -print
```

Note that the name pattern is the ***`basename`***, found in any directory, starting from each of the the ***`starting_dirctories.`*** The ***`basename`*** patterns can include shell-GLOB-style path metacharacters such as "**`*`**" and "**`?`**". Note the unusual use of **full-words** used following *single*-dashes as **options** in this command! (Almost all other commands use *double* dashes for word-style options.) Examples (try them!):

➢ **`find . -print`** *(prints **all** the pathnames under the **current** directory)*
➢ **`find /etc -name 'passwd' -print`** *(print pathnames ending with basename* **`passwd`***)*
➢ **`find /etc -name '*.conf' -print`** *(all pathnames ending in* **`.conf`***)*
➢ **`find /bin -name '?ash' -print`** *(four-character basenames ending in '***`ash`***')*
➢ **`find /lib /usr/lib -name 'lib*.a' -print`** *(multiple starting directories)*
➢ **`find /var/mail -user root -print`** *(print only pathnames owned by this user)*
➢ **`find "$HOME" -mtime -30 -print`** *(print pathnames modified within last 30 days)*

a) What command line recursively **finds** and displays only pathnames owned by userid **`idallen`** under the system directory **`/var/games`** ? (You should see at least one file.)

____

b) What does the **`find`** option "**`-ls`**" do? (See "**`man find`**".)

____

Did you ***READ ALL THE WORDS*** in this Lab?