**FILE DIRECTORY**

Mandatory:

1. Search and display all \*.c and \*.h files in all subdirectories starting from current directory

2. Search and display all directories in all subdirectories starting from current directory

3. Search and display all files with permissions only rw with user and execute permission for others

Optional Assignments:

1. Search and copy all files whose size is < 500 bytes to a directory named “backup”

2. Starting from home directory , find and display all

i. empty files

ii. readonly files

iii. all empty directories

3. Search for a user input pattern in a set of files, extract all lines with the pattern , redirect to a file and display the contents of the file

4. Search and display all files which were modified in last 15 days

5. Search and display all file which were modified in last 10-15 days

6. count the number of words in all \*.txt files in a directory and display

7. In a given directory sort all files based on their size (descending order) and then read and display the contents of each file

**GREP ASSIGNMENT**

Mandatory:

1. Use any .c file. Using grep command extract and display

a. all lines beginning with #include with line numbers

b. display all lines which do not begin with #include

c. display the line number of main()

d. extract all lines containing characters of opening and closing parathesis {(,))

2. Perform the above operations on a set of \*.c files

Optional Assignments:

1. Use a file with email id’s . Use regular expression to extract and display the user and domain names. 2. Obtain the file grepdata.txt. You can save the file on your local system. Once you have the file, write a series of grep statements that do the following:

· Print all lines that contain a phone number with an extension (the letter x or X followed by four digits).

· Print all lines that begin with three digits followed by a blank. Your answer must use the \{ and \} repetition specifier.

· Print all lines that contain a date. Hint: this is a very simple pattern. It does not have to work for any year before 2000.

· Print all lines containing a vowel (a, e, i, o, or u) followed by a single character followed by the same vowel again. Thus, it will find “eve” or “adam” but not “vera”.

**LINUX ENVIRONMENT VARIABLE ASSIGNMENT**

1. State the command for the following.

a. view current environment variables

b. display the PATH evirnment variable value

c. update PATH to include the path to your home directory

2. How will you make the environment changes applicable to all users? Which script is to be modified?

3. I have installed an application named “myapp”. After installation, if I invoke the application, it fails with error as “Command myapp not found”. How will you fix this?

**LINUX ENVIRONMENT VARIABLE ASSIGNMENT**

1. Create a script named “myscript” in current directory to do the following.

a) create a dir named “subd1” in current directory

b) create a file named “dircontent” with contents of the current directory

c) display the contents of the file “dircontent”

d) display the contents of a non existing file named “unknownfile”

2. Run the script and validate the output in following cases

a) Redirect only the stdout to an o/p file named stdout.txt

b) Redirect only the stderr to an o/p file named stderr.txt

c) Redirect both stdout and stderr to an o/p file named stdall.txt

d) Display all o/p and error and also redirect both stdout and stderr to an o/p file named stdall.txt

3. Redirect the output of command below using pipe (|) to wc and get the output .

ls -l

**SHELL SCRIPT ASSIGNMENT**

Mandatory:

1. Create a script named “myscript” in current directory to do the following.

a) display “hello user”

b) list files in current directory

c) name of logged in user

d) date

2. Add execute permission and Run the script from

a) current directory

3. Create a subdirectory named “test”

4. Change to “test” directory. Run the script “myscript” [Hint: You will have to update PATH to include the directory containg script to run script from any location].

Optional Assignments:

1. Considering command line arguments are accessible using $<position> i.e

$0- for first argument

$1- for second argument…

write a shell script to concatenate all command line arguments using ‘\_’ and display it as a single string.

Refer example below

$>./myscript.sh ABC DEF

Output:

ABC\_DEF [Refer http://linuxcommand.org/lc3\_wss0120.php]