OPERATING SYSTEMS - I (2021-22)
EXERCISE-II : Shell Scripts (20%)

Delivery Date: SUNDAY 16/1/2022

1. Write **a script** named **searching** which (a) accepts two integers as arguments and (b) asks the user for the name of a directory, and based on these, displays the following on the screen (1-3 using the *find* command and 4-5 with a combination of the *ls* and *grep commands):* 1. The files of the tree of the given

   directory with authorizations (permissions) the first number (argument) considering it as an octal equivalent.
   2. The files of the tree of the given directory that changed (modify) contents during the last 'x' days, where 'x' is the second number (argument).
   3. The subdirectories of the tree of the given directory that were accessed during the last 'x' days, where 'x' is the second number (argument).
   4. The files in the given directory that all users have read access to.
   5. The subdirectories of the given directory in which they have the right to make changes (create/ rename/delete files) except for the owner and other system users.

   Before printing each list from the above (1 to 5) an appropriate heading should be printed which should mention, among other things, the number of files (or subdirectories) to be printed. The script should be executed repeatedly as long as the user wishes (for different directories) and at the end (before the final output) it should cumulatively display the total number of found (files / subdirectories) of each case (from 1 to 6) for all directories in whom he searched for.

2. In an application we need to create new directories that it will find and use. Write a **script** named **createpvs** that will be called with parameters ROOTFOLDER, no_of_DBFOLDERS, no_of_DATAFOLDERS, USERNAME e.g.

   createpvs /etc/data 0 5 user555

   which will do the following:

   • It will check the number of arguments to be 4
   • It will check if the ROOTFOLDER exists in which the new ones will be created catalogs
   • Will check if user USERNAME exists (in /etc/passwd file)
   • If all checks are successful,
     - It will create as many subfolders named dbfolderÿ inside the ROOTFOLDER as indicated by the number no_of_DBFOLDERS and in such a way as not to "overwrite" existing folders. B.C. if the last folder created by a previous run is dbfolder18 and we want 6 new ones, it should create dbfolder19, dbfolder20,…,dbfolder24 (if the parameter is 0 it should not create any).

     - The same should be true (ie created without “overwrite”) for the datafolderN subfolders specified by no_of_DATAFOLDERS.
     - After creating the folders, it will use the chown command to give ownership of new folders to user USERNAME.

3. Write **a script** named **cmpdir** which compares the contents of two directories (the names of which it will accept as arguments, and it will check if they are indeed directories) in terms of the files they contain. As a result it will initially show for each directory separately, how many and which of its files are not contained in the other directory and what is their total size. It will then display how many and which files are common to the two directories and their total size. Finally, it will move all files shared by the two directories to a third directory (which will also be given/checked as an argument), and create in the two original argument directories appropriate hard links to them.

OPERATING SYSTEMS - I (2021-22)
EXERCISE-II : Shell Scripts (20%)

4. Write a **script** named **bck** that will hold for a specific user (the user name of which will be given as the first argument) backups of one area of its account to another. The script should accept a directory (or file) as the second argument, create a saveable copy of the argument (use **tar)** and copy it to the directory specified by the third argument. However, if the third argument is a file (and not a directory) then it should simply append the copy to be saved to this file. Do the necessary checks for all three arguments (as well as the total number of arguments given during execution).

Appropriately modify the bck script (in **bck1)** to perform the requested backup scheduled (using at), at a specific time of your choice (try to give it as an argument as well).

Modify the bck script (in **bck2)** appropriately so that it runs with no arguments and simply takes a copy of your working directory to /tmp, and schedule (using **cron)** to run every Sunday night at 11pm. for the next six months.

5. Write a **script,** named **mfproc** and able to input 0, 1 or 2 parameters as follows:

Name

mfproc Displays information about processes in the following format:
Name PID PPID UID GID State

Briefly

mfproc [-u username] [-s S|R|Z] -u username

Specifies the user whose processes to display on standard output. If not given, you will display all OS processes

-s state Sets the state of the processes to display on the standard output.
It can be anything between Running(R), Sleeping(S), Zombie

(Z). If not given, then you will display all processes that are in any of the three states

Return value 0 No error
occurred 1 The user
does not exist 2 There is no
process in this state

Note: In the OS proc directory, among others, there are directories named the number of any OS process. More specifically you should use the status file of the /proc/<PID>/status directory. Explanation text for the /proc/<PID>/ status statistics file: Table 1-2: Contents of the status files of the link https://www.kernel.org/doc/Documentation/ filesystems/proc.txt

You don't have to answer all of them to get the maximum score!
Try to do as much as you can!

You should give your answers in a plain text file according to the instructions in the accompanying file 'OS1_lab_guidelines.pdf' (it is posted in Eclass in the same directory as the file of this speech).

Good Success!!!